Author:
Williams, Stephen

Title:
Reduced Order Models for Efficient Gust Modelling

Reduced Order Models for Efficient Gust Modelling

Stephen Peter Ian Williams

Department of Aerospace Engineering

University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctor of Philosophy in the Faculty of Engineering

School of Civil, Aerospace and Mechanical Engineering

August 2018

Word count: 55,396

Abstract

The broad aim of the work carried out was to develop a Computational Fluid Dynamics (CFD) based Reduced Order Model (ROM) capable of the rapid modelling of an aircraft's response to '1-cosine' gusts; for use in identifying critical gust load cases early within the design process where computational resources are at a premium. The first stage of the work involved recreating an already existing Eigensystem Realisation Algorithm (ERA) based ROM; this acted as a base ROM on which developments were made, and performances compared.

The ROM was initially developed and tested on a three-dimensional, inviscid, rigid wing model. As the base ROM showed high accuracy levels (typically comparable to full order CFD results), the focus of development was on reducing the computational cost associated with building the ROM. This was achieved first through reducing the number of inputs needed to create the ROM, and then by minimising the computational cost associated with obtaining that single input. At the end of this portion of the ROM development, the final ROM required just 5.2% of the computational cost incurred by the base ROM.

Further developments and testing were carried out on more complex test cases. For the first of these, a generic wide-bodied aircraft which included the effects of viscosity, the ROM performed as well as it had previously. In the final test case, a simple inviscid, aircraft model which contained both aeroelastic effects and flight mechanics, a slight degradation in performance was noted but was relatively minor.

Finally, modifications were made to the ROM to extend its usefulness. This was achieved by successfully modifying the ROM so that it could be built at one altitude, and then used at another; as long as the Mach number remained constant. Additionally, the ROM was modified so that the surface pressures of the output gust could be constructed; the results of which were generally very good.

Acknowledgements

I would like to take this chance to thank everyone who has supported me in one way or another during my PhD and, in many cases, beyond. Whilst I would love to thank everyone individually, it simply isn't possible; so to all those who have supported me, but whom I am unable to specifically name, a sincere thank you.

I shall start with those who have supported me professionally. **Dr Dorian Jones** and **Dr Ann Gaitonde**, my supervisors; this PhD is a testament to the incredible help and support they have given me throughout. The **Airbus Group**, whose funding and support made this PhD possible; in particular I'd like to thank **AGI** and my industrial supervisors **Mr Martin Herring** and **Dr John Pattinson**.

Then there are my colleagues, some of whom I have shared an office and all of whom I have had many a discussion with (which were critical in keeping me sane at times!). In particular, there are a few individuals whom I would like to single out for their support, which has been indispensable in their own ways; **Dr Christopher Wales**, **Dr Samantha Huntley**, **Mr Amir Bagheri** and **Mr Neil Fraser**.

Finally, there is my family. Starting with the love of my life, **Jayana Finlay**; she has supported, pushed, tested and bettered me. **Michael Williams**, my brother and (rather annoyingly) my best friend; through our shared hyper-competitiveness, he has been one of my main driving forces. Finally there are my parents and role models; **Peter** & **Judith Williams**. I simply do not have the vocabulary to express how much I owe to them, from their unwavering support to the love they have given me throughout my life; this thesis is dedicated to them.

Author’s Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Signed: 

Dated: 17th August 2018

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

ABPOD	Approximate Balanced Proper Orthogonal Decomposition
ARA	Aircraft Research Association
BPOD	Balanced Proper Orthogonal Decomposition
CFD	Computational Fluid Dynamics
Cl	Coefficient of Lift
Cm	Coefficient of Pitching Moment
CSM	Computational Structural Mechanics
DLM	Doublet Lattice Method
DNS	Direct Numerical Simulation
ERA	Eigensystem Realisation Algorithm
FANS	Favre Averaged Navier-Stokes
FFAST	Future Fast Aeroelastic Simulation Technologies
FVM	Field Velocity Method
HHBF	Hicks-Henne Bump Function
LES	Large Eddy Simulations
MAC	Mean Aerodynamic Chord
MIMO	Multiple-Input, Multiple-Output
PDE	Partial Differential Equations
POD	Proper Orthogonal Decomposition
RANS	Reynolds Averaged Navier-Stokes
RBF	Radial Basis Function
ROM	Reduced Order Model
SA	Spalart-Allmaras One-Equation Model
SA-neg	Negative Spalart-Allmaras One-Equation Model
SA-noft2	Spalart-Allmaras One-Equation Model without f_{t2} Term
SA-la	Spalart-Allmaras One-Equation Model with Trip Term
SISO	Single-Input, Single-Output
SST	Shear Stress Transport Model
SVM	Split Velocity Method
ULVM	Unsteady Lumped Vortex Method
VLM	Vortex Lattice Method

Lower Case Latin Alphabet

a	Arbitrary number of matrix rows
b	Arbitrary number of matrix columns
c	Generic constant
d	Distance to nearest wall
e	Generic function
f	Generic function
g	Generic function
h	Mesh spacing
j	Time level; such that $t = j\Delta t$
k	Thermal conductivity
l	Length
m	Number of inputs
n	Size of the unreduced state space vector / rank of the system
p	Number of outputs
q	Heat flux
$\boldsymbol{q}$	Input vector
r	Reduced matrix size
s	Time sampling frequency
t	Time
u, v, w	Velocity along each of the three Cartesian axes (x, y, z respectively)
$\boldsymbol{x}$	State vector
x, y, z	Cartesian coordinates in three dimensions
$\boldsymbol{y}$	Output vector

Upper Case Latin Alphabet

A	System matrix
B	System matrix
C	System matrix
C	Coefficient value
D	System matrix
E	A matrix composed on identity matrices and zero matrices
E	Total energy
F	Flight profile alleviation factor

G	Gust gradient (distance from the start of the gust to the peak)
$\mathbf{H}$	Continuous-time impulse response matrix, also referred to as the Hankel matrix
$\mathbf{I}$	Identity matrix
K	Temperature
L	Characteristic length scale
$\mathbf{M}$	Arbitrary matrix
M	Magnitude of the turbulence vorticity
P	Pressure
$\mathbf{Q}$	A unitary matrix used within Schur decomposition
$\mathbf{S}$	Scaling matrix
T	Time interval
$\mathbf{U}$	An upper triangular matrix used within Schur decomposition
$\mathbf{X}$	Solutions from an impulse (or step-down) response simulation for the primal system
$\mathbf{Y}$	Solutions from an impulse response simulation for the adjoint system

Lower Case Greek Alphabet

α	Angle of attack
β	Number of Markov parameters
γ	Snapshots
δ	The Dirac delta function
ϵ	The rate of dissipation of turbulence kinetic energy
$\boldsymbol{\kappa}$	System kernel
λ	Eigenvalue
μ	Dynamic viscosity
ν	Kinematic viscosity
ρ	Density
σ	Turbulence constant
τ	Viscous shear stress tensor
ϕ	Turbulence parameter
χ	Ratio of turbulent working variable to laminar kinematic viscosities
ω	Wall vorticity

Upper Case Greek Alphabet

Δ	Change in value
$\boldsymbol{\Theta}$	Right-singular values; output of singular-value decomposition such that $\mathbf{M} = \boldsymbol{\Omega\Sigma\Theta}^{\mathbf{T}}$
K	Von Karman constant
M	Magnitude of vorticity
$\boldsymbol{\Sigma}$	Singular values; output of singular-value decomposition such that $\mathbf{M} = \boldsymbol{\Omega\Sigma\Theta}^{\mathbf{T}}$
$\boldsymbol{\Upsilon}$	Simplification representing $e^{\mathbf{A}t}$
$\boldsymbol{\Phi}$	Primal adjoint modes of a system
$\boldsymbol{\Psi}$	Adjoint modes of a system
$\boldsymbol{\Omega}$	Left-singular values; output of singular-value decomposition such that $\mathbf{M} = \boldsymbol{\Omega\Sigma\Theta}^{\mathbf{T}}$

Miscellaneous Symbols

$\star$	Generic variable substitute (used within nomenclature)
$\mathbf{0}$	Zero matrix

Superscript

$\star^{m}$	Modified output system
$\star^{\mathrm{T}}$	Transpose of a matrix
$\star^{*}$	Adjoint of a matrix
$\star'$	Fluctuating component of time-averaged decompositions
$\star''$	Fluctuating component of density weighted average decompositions

Subscript

$\star_{aov}$	Any other value (i.e. non-surface values)
$\star_{c}$	Primal system (typically used in conjunction with γ to denote the number of snapshots in a primal system)
$\star_{ds}$	Design speed
$\star_{F}$	Generic force value
$\star_{g}$	Gust
$\star_{\mathbb{I}}$	Imaginary value
$\star_{i}$	Generic iterator

$\star_j$	Time level j, such that $t = j\Delta t$
$\star_{new}$	New/updated value
$\star_{new_{alt}}$	New altitude ROM is being modified to run at
$\star_{non\_dim}$	Non-dimensionalised quantity
$\star_{ns}$	Total number of eigenvalue shifts
$\star_{old}$	Old/original value
$\star_{orig_{alt}}$	Original altitude of ROM construction
$\star_{pen}$	Penetration
$\star_{\mathbb{R}}$	Real value
$\star_{red}$	A reduced matrix
$\star_{ref}$	Reference value
$\star_{stable}$	A stable (eigenvalue) value
$\star_{sv}$	Surface values
$\star_t$	Turbulent variable (where used, variables without this term refer to laminar variables)
$\star_{trip}$	Value at the trip
$\star_{unstable}$	An unstable (eigenvalue) value
$\star_\infty$	Freestream value

Accents

$\dot{\star}$	Time derivative
$\tilde{\star}$	Discrete form
$\hat{\star}$	Denotes a modified matrix used when creating a ROM from a step-down response
$\breve{\star}$	Denotes a modified matrix used within the restarting process
$\bar{\star}$	Steady state or mean value
$\underline{\star}$	Step-down based system
$\overset{\frown}{\star}$	Density weighted average value
$\overline{\overline{\star}}$	Turbulence working variable

Formatting

Italic only	Standard mathematical variable or value
Bold only	A matrix
Bold & ***italic***	A column vector

1. Introduction

1.1. Background

During their operational life, aircraft will encounter numerous gusts; localised and temporary fluctuations in airflow. These gusts have a large influence in design stages as they are typically responsible for many of the critical cases; where loads are at some of the highest levels in the flight envelope [1]. As such, being able to prove that a given aircraft design can adequately withstand the impact of such gusts is heavily regulated as demonstrated in the Certification Specifications for Large Aeroplanes - CS-25 [2]. These regulations specify a range of gust conditions, both continuous and discrete, which the design must be shown to be capable of withstanding. The work presented in this thesis focuses on the discrete, vertical (relative to the horizontal motion of an aircraft) '1-cosine' shaped gusts specified in the regulations.

Modelling gusts via Computational Fluid Dynamics (CFD) or in wind tunnels is complex; typically resulting in large costs, both financially and in terms of resource-hours. Therefore, these methods are often untenable early in the aircraft design process; where design iterations are typically rapid and often contain substantial changes. As a result simpler methods, which sacrifice accuracy for a reduction in resource-hours, are typically used. However, this is far from ideal as it means early designs have to be conservative, with larger than necessary safety margins applied, to account for the lower accuracy provided by these simpler models. Therefore, whilst there have been numerous efforts to make CFD and/or wind tunnel modelling more feasible during these early design phases, a large opportunity still exists to improve the aircraft design process.

Reduced Order Models (ROMs) created using CFD results could offer a viable solution to the problem. They have been shown to be capable of reproducing gust responses at lower costs compared to standard, high fidelity simulations [3] and so make the possibility of obtaining accurate gust response results feasible during early design phases. Therefore the broad aim of the work carried out within this thesis, was to develop an efficient CFD based ROM capable of rapid modelling of an aircraft's response to '1-cosine' gusts; with an emphasis on those covered within CS-25 regulations.

Some ROMs for gusts have already been developed such as those described by Zaide and Raveh [4] and Wales *et al* [3, 5, 6], and the research presented within this thesis builds on such work. The unique contribution of the work carried out within this thesis is researching and developing more robust, comprehensive and efficient ROMs than previously developed.

1.2. Gusts

Gusts are temporary disturbances to the local airflow and can occur in numerous shapes and sizes. They are heavily related to both wind shear and turbulence, with the differences being more based on direction and discreteness for the sake of easy classification, rather than differences in the atmospheric phenomena. Gusts are typically taken to be discrete changes in airflow, which are smoother than turbulence; although if multiple gusts are encountered in short succession, the effect is extremely similar to turbulence [7].

The causes of these wind related phenomena are varied and can include; changes in terrain (notably mountain ranges), storm-fronts, jet-streams, convectional currents, and more. However, regardless of the cause, the effects are the same and so from a modelling point of view the gust parameters of importance are the shape, magnitude and length.

1.2.1. Significance in Aircraft Design

During their operational life, aircraft will encounter numerous gusts; of all shapes and sizes. These gusts have a large influence in design stages as they are typically responsible for many of the critical design cases; due to the effect they have on both loads and fatigue [1].

During the design process, a gust loads-loop will be carried out to help identify the cases which are likely to be critical cases. This process involves modelling a wide range of gusts, at multiple flight points (taken to be a particular combination of altitude and velocity). From these, a gust loads envelope can be produced which effectively maps out the theoretical maximum loads due to gust encounters. In turn, this envelope is used

to identify the possible critical cases for further, more detailed analysis of the final design. Once the critical design cases are identified, the aircraft structure can be designed to minimise the impact they have on the internal loads and fatigue.

Due to the gust cases often making up a high number of these critical cases, being able to prove that a given aircraft design can adequately handle the impact of such gusts is heavily regulated as demonstrated in the *Certification Specifications for Large Aeroplanes - CS-25* [2].

It can be noted that gust occurrence mid-manoeuvre would in theory produce even more extreme load cases than either regular gust cases or manoeuvres on their own. However, these cases are rarely explored in any real detail due to the extremely low probability of occurrence [8]. This is due both to the small amount of time spent manoeuvring on a typical flight, as well as the ability to foresee gust occurrence in many cases, thus allowing manoeuvres to be delayed until after their occurrence.

1.2.2. Aircraft Certification

CS-25 Regulations [2] require that, in order to be certified, an aircraft must be shown to be able to withstand a range of discrete, '1-cosine' shaped gusts (see Figure 1.1) with corresponding gust gradients (specified to be half the total length of the gust) within the range of 9m to 107m (some aircraft may need to be subjected to additional gusts for certification [2]). It should be noted that the CS-25 Regulations also include certification requirements related to continuous gusts, typically referred to as simply turbulence, however it is the discrete form of gusts that is typically the primary focus of an aircraft loads design process.

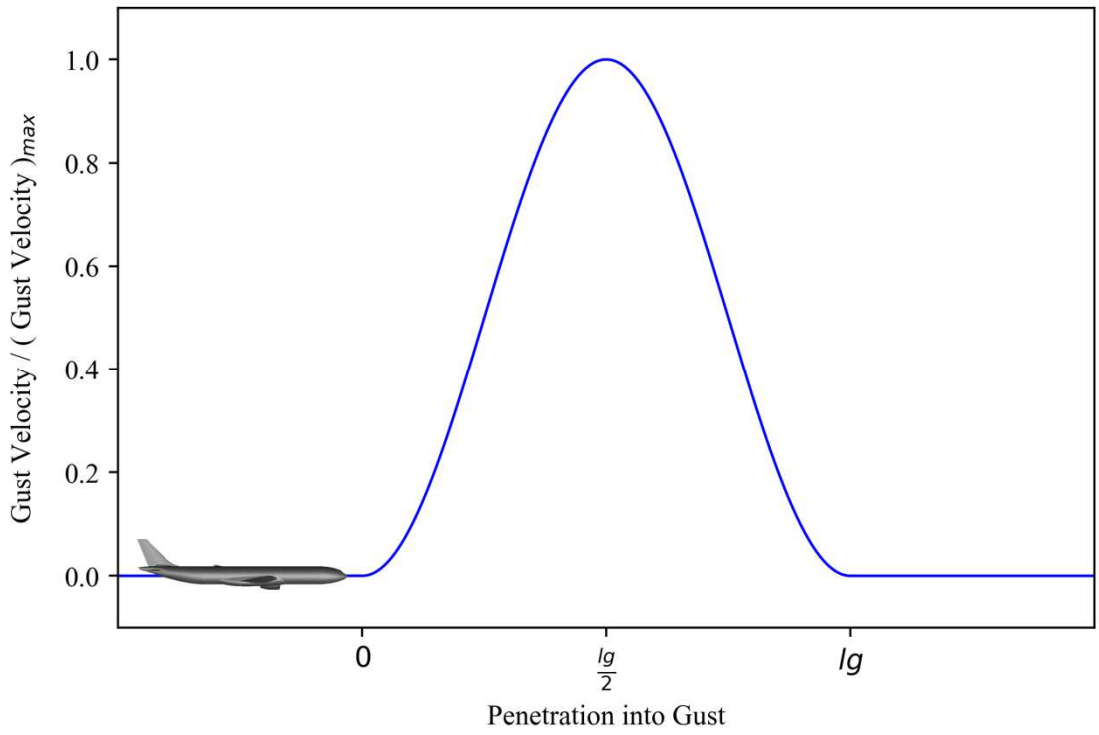

Figure 1.1. Gust velocity normalised by the peak gust velocity, against penetration into gust for a '1-cosine' gust case.

The CS-25 regulations [2] define a '1-cosine' gust as:

$$v_g = \begin{cases} 0, & x_{g_{pen}} < 0, \quad x_{g_{pen}} > \mathrm{l_g} \\ \frac{v_{g_{ds}}}{2}\left[1 - \cos\frac{2\pi\left(x_{g_{pen}}\right)}{l_g}\right], & 0 \geq x_{g_{pen}} \geq l_g \end{cases} \tag{1.1}$$

Where, l_g is the gust length ($l_g = 2G$, were G is referred to as the gust gradient), $x_{g_{pen}}$ is the distance penetrated into the gust, v_g is the gust velocity and $v_{g_{ds}}$ denoting the design gust velocity or peak gust velocity, which is given by:

$$v_{g_{ds}} = v_{g_{ref}} F_g \left(\frac{l_g}{214}\right)^{\frac{1}{6}} \tag{1.2}$$

Where, $v_{g_{ref}}$ denotes the reference gust velocity which varies linearly in accordance to Table 1, and F_g is the flight profile alleviation factor, which is calculated via a number of equations taking into account the maximum aircraft take-off weight, maximum landing weight, maximum zero fuel weight and the maximum operating altitude [2].

Table 1. Reference gust velocities.

Altitude (m)	Reference gust velocity for general cases (ms^{-1})	Reference gust velocity for dive speed cases (ms^{-1})
0	17.070	8.535
4572	13.410	6.705
18288	6.360	3.180

During the certification process it must be demonstrated, using these definitions and equations, that the aircraft design can cope with the loads induced by any discrete gust that the aircraft might be expected to encountered during its operational life.

1.3. Non-CFD Methods of Gust Modelling

1.3.1. Two-Dimensional Numerical Methods

One of the simplest models of unsteady aerodynamics is strip theory. At its core, strip theory is a simple concept; that a wing can be analysed by taking slices along its span, and then treating each strip as a 2D aerofoil (despite having a non-zero span). This method is not without its drawbacks, the primary two of which are that root and tip effects are ignored in the theory, and that the effects of compressibility are also not modelled [1]. Another limitation of strip theory is that significant tapering will cause the predicted lift to be over predicted as it does not inherently account for tapering. However, the base theory was adapted by E. Yates Jr to accommodate variable chord length [9].

The standard strip theory is designed for steady aerodynamics use, however it can be adapted for modelling basic unsteady aerodynamics by combining it with other theories; notably the Wagner [10] and Küssner [11] functions. The Wagner function allows the lift response of an aerofoil to step changes in its angle of attack to be calculated, whilst the Küssner function allows the same type of data to be obtained for a sharp edged gust.

Originally, this Wagner and Küssner based strip theory was only valid for two dimensional flat plates [1]. However, Zaide and Raveh [4] demonstrated that it is possible to extend them to three dimensional wings; although the authors did note that this can only be achieved if the wing is both slender and straight.

Whilst these methods are very basic, with numerous assumptions and limitations, it should be noted that these methods are still used in modern times due to their low computational cost and reasonable accuracy.

1.3.2. Three-Dimensional Numerical Methods

Whilst strip theory is able to handle simple wing geometries, it is unable to be used for more complex, three-dimensional cases. For such cases, a potential flow method, such as a panel method can be used. Instead of splitting the wing into strips along the span, the panel method breaks the surface of the three-dimensional wing down into small plates.

Once the geometry has been broken down into panels, each panel can be represented by a defined distribution of singularities; commonly the distribution of singularity strength is constant or bilinear over each panel. Singularities are mathematical solutions of the Laplace equation that have zero velocity at an infinite distance from the origin, and infinite velocity at the origin itself [12]. The strength of the singularities are then calculated by solving a series of simultaneous linear equations which arise from the boundary conditions.

An alternative, if very similar, method is that of the Vortex Lattice Method (VLM). Like the panel method, VLM splits the model down into a number of panels. However, VLM commonly uses vortex rings on each body centreline panel and horse-shoe vortices in the wake. To model the flow around an aircraft, boundary conditions are used to calculate the strength of each bound vortex; taking into account the effect of induced flow due to neighbouring panels. It is worth noting that VLM can be modified to model gusts by setting the downwash as the gust velocity.

However, a major drawback of both the panel method and VLM is that they are designed for incompressible flow. This can be accounted for to some extent in steady flow using the Prandtl-Glauert correction method; however this method cannot capture unsteady behaviour and strong non-linearities, such as shock waves. Therefore, a loss in accuracy is entailed when compared to fully-compressible models.

In 1969, an alternative method that included the unsteady effects of compressibility, and that was computationally viable at the time, was put forward by Albano and Rodden [13]; the Doublet Lattice Method (DLM).

DLM represents the surface of the model as a set of lifting elements. DLM arises from a linear simplification of the non-linear Euler equations [14]. As a result of the simplification, the number of unknowns being solved is reduced from five in the Euler equations (pressure, density and the x, y, z components of velocity) to just one; the pressure/acceleration potential. Solving for this potential allows for the differential surface pressure to be obtained, using another linear differential equation. DLM allows for complex designs to be relatively accurately modelled, but with a much lower computational cost compared to full order CFD methods. As such DLM is still widely used within the aerodynamic design process of aircraft [1].

Whilst these methods do produce relatively accurate results, they are somewhat limited by being unable to model the effects of viscosity; leading to flow phenomena such as boundary layers and flow separation not being modelled. This means that, whilst able to model the general flow behaviour to a reasonable level of accuracy, they are unable to capture some of the finer details that can be of significant interest.

1.3.3. Physical Modelling

Physical (or experimental) modelling was facilitated by the opening of the first wind tunnel in 1871 by Francis H. Wenham [15]. A wind tunnel creates a completely isolated environment, where an object can be repeatedly and reliably exposed to the same flow conditions to examine its behaviour. It has become relatively straight forward to model steady aerodynamics and/or unsteady cases due to moving structures. However, this has not been the case for gust modelling and remains an area of ongoing development.

The key problem in physically modelling gust aerodynamics lies in the ability to produce a discrete gust. Traditionally, gusts have been produced within wind tunnels via methods such as rotating slotted cylinders [16] or, more recently, active aerofoil(s) (often referred to as vanes) upstream of the test body, which then oscillate to produce flow disturbances [17]. In response to such issues the Aircraft Research Association (ARA) developed a gust generator which utilises fixed vanes with actuated and variable

trailing edge blowing; as laid out by Allen and Quinn [18]. It is worth noting that at the time of their publication they had not yet completed all of their experiments, including only having one of the two planned vanes operational, with its development still in progress. However, the ARA has since carried out some redesign work, have brought both vanes online and are in the final stages of testing [19, 20].

1.4. Computational Fluid Dynamics

Computational Fluid Dynamics is an approach by which the physical conservation equations that govern the fluid flow around an object can be put into a computationally discrete form on a mesh for solution using a range of schemes. With the advent of high performance computers, CFD has been increasingly used in the design of modern aircraft.

The Navier-Stokes equations contain terms for viscous effects, so can model flow separation, and both the Navier-Stokes and Euler equations can model other non-linear phenomena such as shockwaves. Therefore, CFD includes physics not captured in simpler, numerical methods such as the panel method.

1.4.1. Early Developments

In some senses, the development of CFD methods can be traced back to the origins of the finite difference method by Richardson in 1910 [21], and the development of the finite element method by Turner *et al* in 1956 [22]. However, in a more practical sense, the true beginnings of CFD lie in the advancement of computational resources between the 1940s and 1960s. Indeed, whilst it may have been preceded by a few minor forays, the birthplace of CFD is often considered to be the Los Alamos National Laboratory, where the computational resources assembled during the Manhattan Project (and continually developed thereafter) presented the first real opportunity to utilise the power of technology in an attempt to model fluid dynamics [23]. Since then, there have been numerous developments in CFD; with a few of the notable early ones presented here.

Arguably the biggest challenges in the early development of CFD were as a result of transonic flow; with this area of development considered particularly important given

that commercial aircraft cruise at transonic speeds [24]. One such problem was that elliptical partial differential equation (PDE) based algorithms were able to handle subsonic flow conditions but unusable for supersonic ones. Conversely, hyperbolic partial differential equations based algorithms were able to model supersonic flow, but were unsuited to subsonic conditions. This problem was overcome by Moretti and Abbett in 1966 who solved the steady blunt-body problem using a time-dependent, finite-difference method; which made it possible for the hyperbolic, unsteady Euler equations to be solved within the subsonic flow regime [25].

Hyperbolic equations were again at the forefront of CFD development in 1969, when the self-titular MacCormack method was developed for discretising and solving such partial differential equations [26]. The MacCormack method can be considered to be a development upon the Lax-Wendroff method, first put forward in 1960 [27], with the most notable improvement being that the MacCormack method is typically easier to program. It is for this reason that the method is still used today.

Later, in 1970, Murman and Cole demonstrated a method for obtaining stable solutions to transonic cases by using a combination of central differences for the subsonic regions and upwind differencing in the supersonic regions [28]; highlighting that such simulations weren't necessarily computationally prohibitive [29]. Other important developments from around this time include the development of a method to model aerofoils and wings in transonic conditions by Jameson in 1974 [30]; which numerically proved the existence of the weak solutions to the potential equation if a valid entropy inequality is applied. Later, in 1977, Brandt built on the work by Bakhvalov [31] and Fedorenko [32, 33] in the 60s in order to demonstrated large computational savings by blending the processes of discretisation and solving the flow [34]; in doing so pioneering the development and use of multi-grid methods as efficient ways in which to solve elliptical boundary-value problems amongst others [35].

In 1984 Jameson, along with Baker and Weatherill, again contributed a key development to CFD by modelling an inviscid, transonic flow over a whole aircraft model by, amongst other things, pioneering the use of unstructured meshes [36]. Over the following decade unstructured meshes became increasingly popular, particularly for more complex problems. This lead to Simon, in 1991, demonstrating a method by which unstructured meshes could be partitioned for parallel processing [37].

However, as the relative computational cost of gust simulations was, and arguably still is, extremely high, the development of gust response CFD codes was relatively muted until the late 1990s. Since then one of the key problems facing CFD based gust modelling, is the problem of dissipation of the gust as it travels through the domain.

1.4.2. Propagation of Flow Disturbances

One of the problems facing CFD codes is the dissipation of flow disturbances; such as gusts or vortices as they transit through the flow domain; in both cases, the problem typically stems from numerical dissipation. This often leads to the necessity of a fine mesh to avoid the dissipation of such features as they traverse the domain when introduced via a far-field boundary condition; this results in a prohibitively high computational cost. It is potentially possible however to reduce such computational drawbacks via a number of methods.

One possible method was demonstrated by Tang and Baeder in their 2007 journal paper [38] in which they ran a mesh generator along with the CFD solver to locally refine the mesh as a vortex traversed the computational domain. Due to the similar way in which gusts and vortices are dissipated within CFD simulations, this method could, in principle, be applied to gust cases; with the mesh being refined in the region of the domain that contains the gust at any given point. This approach, however, adds complexity to the simulation process and can make it cumbersome to implement. Additionally, unlike vortices, gusts cover a very large area of a given computational domain (at times over 200 meters long and the entire width of the domain) and refinement on this scale becomes just as computationally expensive, if not more so, than simply having a fine mesh to begin with. Another technique that can be utilised is the application of a chimera grid/mesh approach (also known as an overset grid/mesh approach) developed by Steger *et al* in 1983 [39]. This overlays two or more meshes and thus a finer mesh can be applied locally to the gust. However, again, this adds complexity to the process; most notability in the compiling of the results in the region where the two grids overlap. It is also possible to employ aeroacoustics techniques to avoid dissipation of the gust, as shown by Hixon *et al* in 2006 [40], however the applications of such techniques from an industrial standpoint are often limited due to the

rarity of the employed schemes within the CFD programs used in the relevant design processes.

Perhaps the most notable advancements in the CFD simulation of gusts in terms of overcoming this dissipation problem was the development of the Field Velocity Method (FVM), also known as the artificial velocity method, in 1997 by Baeder and Parameswaran [41] and Singh [42, 43]. FVM breaks down the flow field velocity into the gust velocity and a remainder. The gust velocity, along with its motion, is prescribed, which eliminates the possibility of the gust being dissipated and leaves just the remainder to be solved for. As such FVM offers two key benefits over other methods. The primary benefit is a large computational saving by removing the necessity to have either a fine mesh away from the areas of interest near the aircraft, or to have additional meshing complexities such as local mesh refinement. The secondary benefit is that the method is relatively simple, as it effectively imposes a velocity without mesh movement, and can be directly applied within almost any moving mesh CFD code; thus allowing it to be adopted within industrial setups with minimal modification required. FVM however does have one flaw in that it neglects terms in the Euler / Navier-Stokes equations which result in it not capturing the effect of the aerofoil/aircraft on the gust; thus preventing it from accurately modelling aeroelastic problems where these terms may not be small.

This limitation of FVM was overcome by Wales *et al* [44] with the development of the Split Velocity Method (SVM). The SVM is very closely linked to the FVM, with the general principles behind the two methods being the same. However, where they differ is that the SVM does not simplify the flow equations and thus retains the source terms necessary to capture the effects of the aerofoil on the gust.

1.4.3. Validation of Results

Validating the results from a CFD simulation is a relatively complex procedure. Before considering the process of validation, it is important to make the distinction between validation of results and verification of the code itself. The latter refers to the process by which the code is checked for mathematical or programming errors, by comparing the output to exact analytical solutions [45]. Conversely, to validate a set of results, a

comparison must be made between the results of a simulation and those of real world experiments.

One of the complications that arises from validating results from a CFD simulation is that it is not possible to universally validate all results [46]. Instead, the results can only be validated for a specific set of conditions; outside of which further validation would be required. Another consideration is that the effects of aeroelastic deformation are important. This is highlighted by Heinrich *et al* [47] who demonstrated that when modelling a 2.5g flight case on a 1g flight mesh the root bending moment was significantly higher than when modelling on a 2.5g flight mesh. This emphasises that if a CFD model fails to take into account aeroelastic deformation, the results would not match those of a wind tunnel model that is free to deform.

Once the conditions for validation are established, the process itself is relatively straight forward. Firstly the results should converge onto a solution, and that solution should be checked to show it upholds expected flow behaviour (such as mass conservation, etc.). After this, the results should be checked to be independent of both mesh density and time step size (if not a steady solution). Finally, the results must be directly compared to real world experiments, typically wind tunnel testing, to check that the results match; during this process, various turbulence models should be explored to see the impact they have on the CFD results. It should also be noted that it is exceptionally difficult to produce a discrete gust in a wind tunnel; which further complicates the validation process as a like-for-like comparison is not always possible.

1.5. Reduced Order Models

Reduced Order Models can be thought of in general terms as a method by which a model can be simplified to obtain a new model which is computationally less expensive but still provides a good approximation of the original model. Often this is done by reducing the complexity of the model through approximations and assumptions. Many developments in ROMs have taken place in fields such as control theory, and only more recently has focus on fluids increased.

1.5.1. A Brief History of ROMs

The origin of ROMs is hard to define due to its relatively abstract concept; with different authors assigning different meanings to the term. If considering ROMs at their most basic level then, for example, the Euler equations could be considered as a reduced order version of the Navier-Stokes equations. A more widely used definition of what constitutes a ROM is perhaps: a ROM is the outcome of a Model Order Reduction technique that is used to capture the dominant effects of a high fidelity model, whilst greatly reducing the computational cost. Using this definition, then perhaps one of the earliest pioneers is Moore, who in 1981 developed a method that utilised Singular Value Decomposition (SVD) to produce a variation on Kalman's minimal realization theory, that did not suffer from the instabilities that the original was prone to [48].

Another pioneer of ROMs is Karpel, who in 1982 developed an active control system for gust alleviation and flutter suppression [49]. This system was built around a state-space model of the aeroelastic behaviour of an aircraft, which allowed for real time approximations of the loads to be made. Karpel continued to develop further methods for designing ROMs and demonstrating their application, such as his development of a dynamic realization based aeroelastic ROM in 1990 [50], and optimisation processes that were built around ROMs in 1992 [51] and 1999 [52].

In 1994, Hall demonstrated a method for building ROMs by using the eigenmodes of the system being modelled [53]. He applied this to a series of unsteady aerodynamic models, both two-dimensional and three-dimensional, before highlighting potential applications for the ROMs; such as active control methods for aeroelastic behaviour. Hall (along with Florea and Lanzkron) would also notably build on this work in 1995 by developing a ROM for the unsteady aerodynamics within turbomachinery cascades by using eigenmodes and eigenfrequencies produced by a Lanczos method [54]. Around the same time, Gallivan, Grimme, and Van Dooren developed a method that used Lanczos methods to produce Padé approximations of a system [55]. In 1996, Dowell also developed a ROM for unsteady aerodynamics which was built around the Lanczos method to obtain the eigenmodes of the system.[56]

In 1995, Romanowski developed an unsteady aerodynamic ROM based on the Euler equations [57]. To achieve this, Karhunen-Loève decomposition was used to obtain the

eigenmodes of the system from which the ROM was built. The following year, Romanowski, along with Dowell, developed and compared two unsteady aerodynamic ROMs built around the Euler equations [58]. One of the ROMs was built using Karhunen-Loève decomposition, whilst the other was built using a modified WYD (so named after the original authors Wilson, Yuan and Dickens [59]; and also referred to as Ritz-Wilson) reduction. They found that both produced extremely accurate results, however on fine meshes the WYD method tended to converge on the required eigenvalues quicker than the Karhunen-Loève method.

Another key development in ROMs was achieved by Kim in 1998 [60]. Kim developed a method for using the Karhunen-Loeve procedure to obtain the eigenmodes of linear systems within the frequency domain. This allowed for a frequency domain based ROM to be created which he noted to be extremely comparable to the equivalent time domain methods; with the note that he expected the frequency domain version to be notable better at handling more complex systems.

Proper Orthogonal Decomposition (POD), which builds on the work of Karhunen [61] and Loève [62] in the 1940s and 50s, has been extensively used in the modelling of fluids; both in the time [63–65] and frequency domains [60, 66, 67]. However, it was not until 2002 that Willcox and Peraire first demonstrated a method using POD to construct a balanced ROM which considered the system outputs as well as the inputs; this method was demonstrated on the unsteady aerodynamics of a two-dimensional aerofoil [68]. They noted that the method could be used in either the time or frequency domains and yielded highly efficient ROMs. Since then, POD has become widely used within the construction of ROMs [69, 70], and has also been further developed to create a method known as Approximately Balanced Proper Orthogonal Decomposition (ABPOD) [48, 68, 71, 72]. More recently, in 2018, Bekemeyer *et al* [73] put forward an unsteady, non-linear physics based ROM using POD; this marked the first time such a ROM was applied to industrially relevant aircraft test cases.

Another method for constructing ROMs is the use of the Eigensystem Realisation Algorithm (ERA); which was originally developed in 1985 by Juang and Pappa [74]. This method was first used in the creation of ROMs by Silva and Raveh in 2001 [75] and has become increasingly popular [3, 76–79]. In 2011 Ma *et al* [80] demonstrated that the same ROM can be produced using either ABPOD or ERA. This marked a big

breakthrough in the development of ROMs as those created using ERA do not require adjoint data; thus greatly reducing the computational cost when compared to ABPOD.

1.5.2. Methods of Modal Identification

Whilst there are numerous methods of modal identification, many ROMs use Proper Orthogonal Decomposition, approximately balanced Proper Orthogonal Decomposition or Eigensystem Realization Algorithm.

POD can trace its roots back to the independent work of two mathematicians, Karhunen [61] and Loève [62] which formed the basis of what is referred to as the Karhunen-Loève theorem. This theorem formed the basis of POD as the work was developed further by numerous other researchers. However, of all those who worked on developing POD, perhaps the most noteworthy is Lumley who it could be argued truly pioneered its application within fluid dynamics [65]. POD has proven a popular choice over the years for modal identification within ROMs [69, 81], particularly as it can be computed directly from experimental data or computational simulations; although it should be noted that this is dependent on the number of snapshots (effectively this is the full solution at a single instance in time) you require. However, it is not without flaws, and Ma *et al* [80] highlighted how the method can be unpredictable and at times the performance unsatisfactory. Perhaps the biggest drawback of POD is that it finds the high energy modes which are the most representative of a given dataset; however this is not always desirable. Indeed, it is often more desirable to have a method that finds both the high energy modes, but also finds the modes that are most likely to influence the future dynamics of the system; something that is achieved by ABPOD [70].

ABPOD, which is often referred to as just Balanced POD (BPOD) despite only being approximately balanced, is hard to attribute to one source, with Moore [48] arguably being the first to begin work on developing such a method, followed by key developments by Lall *et al* [71, 72], Willcox & Peraire [68] and Rowley [70]. The method combines elements of both POD and balanced realization theory. The new method was noted by Willcox and Peraire [68] to produce much improved models when compared with the original POD technique. To achieve this though, the method requires data from adjoint simulations. This adjoint data causes the method to have a few drawbacks. One of which is the inability for a ROM employing ABPOD to be built

from experimental data. Whilst this is not an issue for CFD based ROMs, they have their own drawback with additional simulations increasing the computational expense.

ERA was developed by Juang and Pappa in 1985 [74], building on the work of Kung [82]. Unlike ABPOD, ERA does not require adjoint data but is a balanced method. Thus, the method can be applied to experimental data. Additionally the method is less computationally expensive (although, like ABPOD this will be dependent on a number of factors). Ma *et al* [80] found in their experiments that ERA produced a ROM that was near identical to those obtained via ABPOD. Indeed, the authors noted that in some circumstances ERA can produce ROMs of a higher accuracy than ABPOD owing to small inaccuracies within the adjoint operator Thus, it is clear that ERA is an excellent method for modal identification in the construction of a ROM. Unless adjoint information is available, in which case ABPOD does offer some small advantages; namely that it produces more versatile modes due to the method capturing more of the physics of the system. Additionally, it should be noted again that when pulse impulses are used, ABPOD and ERA produce the same ROM [80].

1.5.3. Non-Linearities

Perhaps the most notable methods of modelling non-linear behaviours is Volterra theory (also known as Volterra series); which is a mathematical process which allows the nonlinear behaviour of a given system to be captured when using a complete time history of a system response [83]. The theory dates back to the work done by Volterra in the late 1800s, and was notably built on by Weiner during the Second World War for the application of the theory for system analysis [84], which opened the possibility of applying it to nonlinear system analysis.

In the continuous-time domain (see Section 3.1.1), Volterra theory effectively represents a non-linear system as an infinite sum of multi-dimensional integrals of increasing order. Within the discrete-time domain the same approach is applied, but with the integrals changed to discrete summations.

It was Silva who pioneered its use within nonlinear aerodynamic modelling, first with his work on applying it to transonic nonlinear aerodynamics in 1993 [85, 86], before then further developing it with respect to impulse response identification in 1997 [83,

87] and culminating in the application of the theory to the development of ROMs [88]. In these works the author used Volterra theory to identify system kernels based on the aerodynamic system response to an impulse response.

System kernels can be thought of as being the properties of a modelled system, broken down into an infinite number of multi-dimensional convolution integrals [89]. This infinite set of integrals are of increasing orders and if summed would perfectly model the system. However, in practice this infinite series can be truncated to a handful of the lower order kernels, and still adequately capture the system. For example, the behaviour of weakly nonlinear systems can be captured to a reasonably high level of accuracy by just the first two terms in the infinite series; the zeroth and first order kernels, which are the linear pulse response and the nonlinearity modelled by the effect of a delay on a pair of pulse reponses.

Raveh [90] demonstrated that whilst Volterra theory can be used to identify second order kernels, or indeed higher orders ones, their calculation requires a sharp increase in computational expenditure and fails to provide any significant improvement in the accuracy of the ROMs for which they were applied. The author also compared ROMs build around first and second order kernels identified via Volterra theory to a ROM developed by Raveh *et al* [91] where the step response of the system served as its kernels instead. They found that the Volterra kernel based ROMs were highly sensitive to the parameters of the impulse response, leading to sharp responses causing inaccuracy in the results. This problem was not present in the step response based ROM, giving the method a clear advantage over more traditional Volterra based ones. However these results should be viewed with caution as the time step used for the "velocity component" of the pulse response means the assumption of linearity was invalid.

In instances such as shocks where non-linearities cannot be, or are not, linearised, the accuracy of the results can suffer. A clear example of this is highlighted by Wales [92] who identified that their results were being negatively affected by the development of a shock over the upper surface of their test wing as a '1-cosine' gust traversed it; when previously the surface had been free of a shock. Whilst there have been some attempts to develop shock capturing, such as by Lindquist and Giles [93] and by Amsallem and Farhat [94] it still presents a notable challenge for building ROMs.

1.5.4. Discrete and Continuous Time Domains

An interesting aspect of CFD based ROMs is the way in which they progress time. CFD cannot be solved in the continuous time domain, and thus is implemented within a discrete time domain. As a result, CFD based ROMs are often built within this domain. This is not ideal, as it limits the application of such ROMs to problems with the same time step size as the original CFD. This is an important restriction in terms of the application of ROMs within aeroelastic simulations as they cannot then be coupled to discretely non-linear structural models, as highlighted by Gaitonde and Jones [95]. Logically, it makes sense to attempt to build a continuous time ROM from a respective discrete time version. This poses a key problem, as highlighted by Wales *et al* [6] in that the new ROM may have stability issues (see Section 1.5.5); irrespective of the stability of the discrete time parent ROM.

1.5.5. ROM Stability

When producing a ROM, it is important that the system matrices are stable. This stability is defined, for a discrete system, as the system matrix $\widetilde{\mathbf{A}}$ (see Section 3.1.2) having no eigenvalues with an absolute value greater than 1 (see Figure 1.2). Should an eigenvalue exist with a value greater than 1, then the system will experience an undesired behaviour such as an exponential growth and/or an oscillation. Therefore, it is imperative when building a ROM that all eigenvalues fall within this stable region.

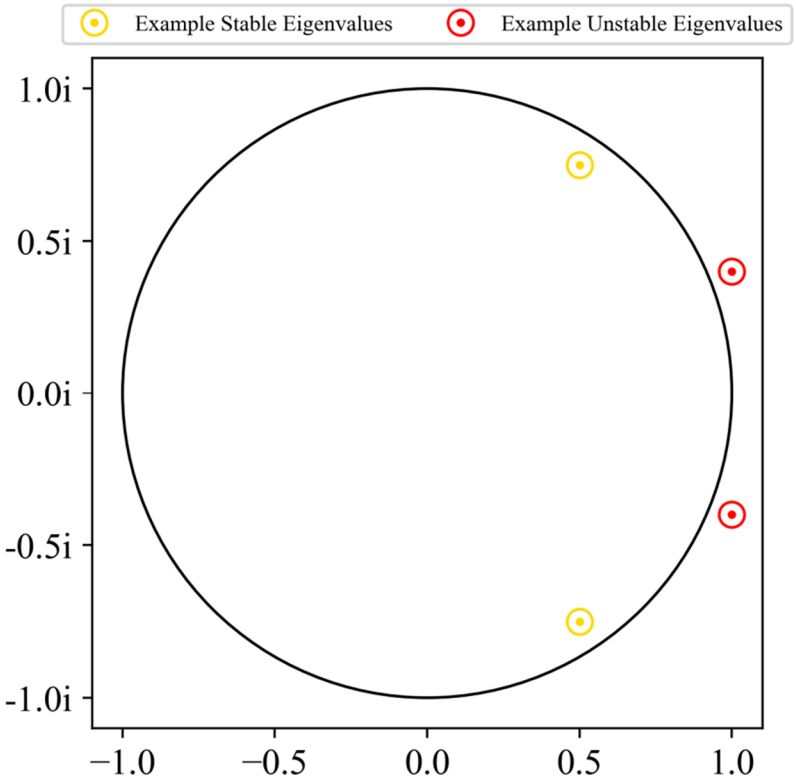

Figure 1.2. Stability region for discrete-time based ROMs.

To ensure the ROM built is stable, a technique known as restarting can be used. The technique was first developed for the Lanczos and Arnoldi methods by Grimme *et al* [96] and Jaimoukha *et al* [97] before being demonstrated for ERA by Wales *et al* [6].

Restarting takes the eigenvalues produced by the ROM and checks if any of the eigenvalues have a magnitude greater than 1 for a discrete system. If this is the case, then the system is deemed unstable and restarting is carried out; otherwise the system is stable and the ROM construction is completed. If restarting is carried out, a polynomial transformation is used where the roots are placed where the unstable eigenvalues occur. This allows a new set of eigenvalues to be produced, without the previously identified unstable ones present.

One consideration that must be made when implementing restarting is how many unstable eigenvalues to process during each restarting cycle. This was explored by Wales *et al* [6] who noted that, whilst possible to process multiple unstable eigenvalue per restart cycle, slightly more accurate ROMs were yielded when only one unstable real eigenvalue (or complex pair) was dealt with at a time. Similarly it also noted that the choice of which unstable eigenvalue to process in any given restart cycle did not

make much difference, choosing the one further removed from the origin [0,0] did increase accuracy slightly.

Whilst restarting is perhaps the most common method by which ROM stability can be assured, there are alternatives. McKelvey *et al* [98] put forward a method which utilises Schur decomposition to allow the unstable eigenvalues to be projected back within the stable region. Extremely unstable eigenvalues (with an absolute value greater than 2) are set to zero, eigenvalues with an absolute value of exactly 1 have a small value subtracted from the magnitude to move them within the stable region, and all other unstable eigenvalues are effectively reflected back into the stable region. All the unstable eigenvalues are processed at once, and this new set of eigenvalues is then used to reverse the Schur decomposition to produce a stable $\widetilde{\mathbf{A}}$ matrix.

1.5.6. Method of Implementation

Ultimately the primary aim of CFD based gust ROMs is to reduce the resource requirements, both in terms of computation and hands on worker time, for obtaining the aircraft response to a gust. However, it is extremely rare for potential resource savings in the way the ROM is applied (for example, how the code is run) to be explored. This can be the case for numerous reasons, and does not detract from the work done by those researching ROMs, but it does mean there is likely to be opportunities to expand upon some already existing methods to further reduce their computational expense.

One example of an author detailing their full process is Wales [99]. Firstly, three CFD simulations are carried out via the TAU-code [100]; one a combined series of steady simulations at various angles of attack, and two sharp-edged gusts with one magnitude double the other. The results of these simulations must then be manually modified to ensure they can be read by MATLAB [101]. A MATLAB code is then applied to convert the sharp edge responses into pulse response files. Then two further MATLAB codes, using the data from both the aforementioned pulse calculations as well as the CFD data, are executed; with one code calling the other as necessary. This outputs the ROM in terms of relevant matrices and variables which are read by a final MATLAB code which calculates to response to any given gust.

The aforementioned process is fairly typical of the execution of techniques where the focus is not on their industrial application. The author is successful in achieving what they set out to do, however if all the elements of the ROM were combined into one package, then it is possible that the computational cost could be further improved via various changes and modifications to the process.

The first type of modification possible arises due to the fact that when breaking any process down into smaller sub-processes, you also run the risk of introducing duplication and/or redundancy. This can bloat a process and inflate the computational cost by repeating steps more regularly than would otherwise have been required.

The second type of change that can be made, which could be considered a sub-section of the first but is worth highlighting individually, is demonstrated in the aforementioned process by Wales [99] which involves writing to disk and then immediately reading back in the same data. In some instances, this could save computational resources; such as if computing the system matrices requires more computational cost than the write/read process. However, often this will add unnecessary overhead, slowing down the processes and tying up computational resources for longer than actually required.

Finally, the third main type of change that can be made is more subtle that the previous two. When a process like building a ROM is broken down into multiple, separate processes, it can become hard to get an intuitive feel for how changes impact the computational cost of building the ROM and it can also become hard to spot areas of inefficiency. Finding these inefficiencies and keeping an eye on the impact changes have on the computational cost are both critical to ensuring the implementation is optimised.

1.5.7. Validation of Results

For a ROM built using CFD, the best result possible is a perfect reproduction of the results that can be produced by that CFD code. Thus they can be, and typically are, compared directly to the results of full CFD gust simulations. This gives research into ROMs a distinct advantage when compared with the development of new CFD code based techniques in that wind tunnel tests, which can be complex, time consuming and expensive, are not required.

However, an important, yet easy to overlook, aspect of ROMs is the concept of only getting out what has been requested. The very nature of ROMs means that the information about the model in question is being reduced. This allows for large computational savings to be made, but also means that a new ROM must be created if the desired outputs change after its original creation.

1.6. Summary of Motivations and Aims

1.6.1. Summary of Motivations

The effects of gusts within the design process cannot be understated (see Section 1.2). Gusts represent many of the critical design cases for aircraft loads, and so being able to model them with a high level of accuracy is desirable as they heavily influence the wing structure design. Ideally, a high accuracy loads-loop would be carried out on each iteration of the aircraft design, but this isn't always possible.

The complexity of modelling gusts in CFD or wind tunnels means that it is still common for more traditional, simpler mathematical models to be used early in the aircraft design process (see Section 1.3). This continued reliance on simpler method highlights how a balance must be made between accuracy and cost (computational, financial and/or time). Certainly, early in the design process when changes are often rapid and substantial, the cost of using CFD or wind tunnels as standard for gust modelling becomes sufficiently large as to become less attractive an option than using simpler, less accurate methods, and then simply incorporating an additional safety margin. Therefore it is possible to make a notable improvement to the design process of aircraft should a method of modelling gusts be developed that is highly accurate but computationally inexpensive.

Whilst ROMs date back to around the 1980s (see Section 1.5), it is rare to find a case where a ROM has been tailored to a particular application; instead they are typically more generic, proof of concepts. As such, there exists room for a unique and substantial contribution to be made.

1.6.2. Summary of Aims

The broad aim of the work laid out in this thesis is to take an already existing methodology for constructing a ROM, this baseline ROM method is used to build and test the ROMs in order to improve it with respect to its potential use of modelling gust cases within a loads-loop; although other applications are considered. The baseline ROM method is one already established and serves as a benchmark against which all developments can be compared.

Initially a 3-Dimensional rigid wing, in inviscid flow conditions, is considered. This case allows for a detailed look at how the baseline ROM method performs, and from which developments can be made. The modifications made to the baseline ROM method improve it in some meaningful way; either in accuracy or computational cost (whichever would most improve its suitability in its primary application).

Once the key developments of the ROM are complete, the latest version is applied to an aircraft model in viscous flow conditions. This case is used to, where applicable, verify the previous developments of the ROM and explore how the ROM performs with a more representative model. Where applicable, further improvements to the ROM are made at this point.

After the ROM has been sufficiently tested on the aircraft model, a final test model is introduced. This model is a simplified aircraft model consisting only of a wing and an empennage. Whilst the model is simpler from a geometric standpoint and does not include the effects of viscosity, it includes both aeroelastic deformation and allows for flight mechanics to be modelled. As a result of the inclusion of flight mechanics it can still be considered an extreme test case that goes beyond what might be reasonably expected within a loads-loop process. This allows for greater response of the aircraft but as there is no gust alleviation it may produce responses not seen in flight. This case will be used to explore the limits of the ROM and identify at what point it starts to experience degradation in accuracy.

Finally, adaptations to the final ROM will be explored. These adaptations effectively sit on top of the ROM (in that they could be disabled with no effect on normal use) and improve its use in some meaningful way.

2. Overview of Governing Equations and Flow Solver

The behaviour of a compressible Newtonian fluid is governed by the set of equations derived independently and almost simultaneously by Claude-Louis Navier and Sir George Stokes, the Navier-Stokes equations [102]. These consist of a set of equations for conservation of mass, momentum and energy, together with equations of state. These equations cannot be solved analytically for the flow about aerospace geometries and it is therefore necessary to make simplifying assumptions or resort to numerical solution processes, or a combination of these two approaches. CFD has therefore been the focus of considerable research effort over the last 50 years and many different methods has been developed. Since the aim of this thesis is to build efficient ROMs of the high order CFD, the governing equations and an overview of flow solvers is presented in this Chapter along with more detail on the DLR TAU code used here.

2.1. Three-Dimensional Differential Navier-Stokes Equations

The Navier-Stokes equations for a compressible Newtonian fluid in differential form are given in terms of dimensional variables by:

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial x}+\frac{\partial \rho v}{\partial y}+\frac{\partial \rho w}{\partial z}=0 \tag{2.1a}$$

$$\begin{aligned}
\frac{\partial \rho u}{\partial t}+\frac{\partial \rho u^2}{\partial x}+\frac{\partial \rho uv}{\partial y}+\frac{\partial \rho uw}{\partial z} &= -\frac{\partial P}{\partial x}+\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}+\frac{\partial \tau_{zx}}{\partial z} \\
\frac{\partial \rho v}{\partial t}+\frac{\partial \rho vu}{\partial x}+\frac{\partial \rho v^2}{\partial y}+\frac{\partial \rho vw}{\partial z} &= -\frac{\partial P}{\partial y}+\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y}+\frac{\partial \tau_{zy}}{\partial z} \\
\frac{\partial \rho w}{\partial t}+\frac{\partial \rho wu}{\partial x}+\frac{\partial \rho wv}{\partial y}+\frac{\partial \rho w^2}{\partial z} &= -\frac{\partial P}{\partial z}+\frac{\partial \tau_{xz}}{\partial x}+\frac{\partial \tau_{yz}}{\partial y}+\frac{\partial \tau_{zz}}{\partial z}
\end{aligned} \tag{2.1b}$$

$$\begin{aligned}
&\frac{\partial \rho E}{\partial t}+\frac{\partial u(\rho E+P)}{\partial x}+\frac{\partial v(\rho E+P)}{\partial y}+\frac{\partial w(\rho E+P)}{\partial z} \\
&\quad=-\frac{\partial q_x}{\partial x}-\frac{\partial q_y}{\partial y}-\frac{\partial q_z}{\partial z}+\frac{\partial u\tau_{xx}}{\partial x}+\frac{\partial u\tau_{yx}}{\partial y}+\frac{\partial u\tau_{zx}}{\partial z}+\frac{\partial v\tau_{xy}}{\partial x} \\
&\quad+\frac{\partial v\tau_{yy}}{\partial y}+\frac{\partial v\tau_{zy}}{\partial z}+\frac{\partial u\tau_{xz}}{\partial x}+\frac{\partial u\tau_{yz}}{\partial y}+\frac{\partial u\tau_{zz}}{\partial z}
\end{aligned} \tag{2.1c}$$

Where ρ is the density; u, v, w are the velocity components in the Cartesian coordinate directions x, y, z; P is the pressure; q is the heat flux; E is the total energy and τ_{ij} is the viscous shear stress tensor. The equation for pressure is given by:

$$P = (\gamma - 1)\left[\rho E - \frac{\rho(u^2 + v^2 + w^2)}{2}\right] \tag{2.2}$$

where γ is the ratio of specific heats, and the heat flux is given by Fourier's law as:

$$q_j = -k\frac{\partial T}{\partial x_j} \tag{2.3}$$

where k is thermal conductivity, T is temperature and the subscript $j = 1,2,3$ is used to indicate each of the respective Cartesian coordinate directions. The shear stresses are given by:

$$\begin{aligned}
\tau_{xy} &= \tau_{yx} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \\
\tau_{yz} &= \tau_{zy} = \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) \\
\tau_{zx} &= \tau_{xz} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\
\tau_{xx} &= \mu\frac{2}{3}(\nabla.\boldsymbol{u}) + 2\mu\frac{\partial u}{\partial x} \\
\tau_{yy} &= \mu\frac{2}{3}(\nabla.\boldsymbol{u}) + 2\mu\frac{\partial v}{\partial y} \\
\tau_{zz} &= \mu\frac{2}{3}(\nabla.\boldsymbol{u}) + 2\mu\frac{\partial w}{\partial z}
\end{aligned} \tag{2.4}$$

where μ is dynamic viscosity. Then for laminar flow the dependence of viscosity on temperature would be given by Sutherland's law as

$$\frac{\mu}{\mu_\infty} = \left(\frac{T}{T_\infty}\right)^{3/2}\left(\frac{T_\infty + 110}{T + 110}\right) \tag{2.5}$$

where the subscript infinity symbol denotes a freestream value.

However, most viscous flows cannot be considered laminar due to turbulence. Capturing the effects of turbulence is therefore an important aspect of CFD and this can be done in a number of ways. The most appropriate approach depends on the specific nature of the flow to be simulated.

2.2. Turbulence Modelling

2.2.1. Direct Numerical Simulation

The governing Navier-Stokes equations can capture turbulence without modification, a method known as direct numerical simulation (DNS); however in most engineering

situations the computational cost is prohibitively large and alternative approaches are used. This cost results from both the very fine mesh needed to capture the smallest disturbances and the extremely small time steps needed to simulate the fastest occurring eddies.

2.2.2. Large Eddy Simulation

One approach to reduce the computational cost of simulating turbulent flows is to use a low-pass filter of the Navier-Stokes equations. Only the largest turbulent eddies are then resolved as part of the simulation and the effects of the smallest eddies are instead modelled; this method is known as large eddy simulation (LES). This is a reasonable approach for many flows as the smallest turbulent eddies (those which are smaller than the refinement of the mesh and/or those which occur in less time than a single time step) carry only a small amount of the turbulent energy and are universal in nature. The larger eddies in contrast carry most of the turbulent energy and are dependent on the geometry over which flow is simulated. LES can be used with significantly less grid points than a DNS simulation, however at the Reynolds numbers typical of aerospace applications the cost is still prohibitive for most investigations.

2.2.3. Reynolds and Favre Averaged Navier-Stokes Equations

The cost of methods such as DNS and LES is high, even at the lowest Reynolds number (Re) for most geometries, however the costs increase rapidly as the Reynolds number increases. Thus, at the high Reynolds numbers encountered in aerospace applications these methods are of limited use. Instead the high computational cost of modelling flow turbulence is overcome in CFD simulations by using averaging methods. For incompressible flow simple time averaging can be used. This splits any flow variable in the Navier-Stokes equations into a mean and fluctuating component, with the mean component defined by averaging over a time period longer than the turbulent scales. Thus if ϕ is a turbulent variable then

$$\phi(x, y, z, t) = \bar{\phi}(x, y, z, t) + \phi'(x, y, z, t) \tag{2.6}$$

Where t represents time, the over bar denotes the time-averaged component, and the accent denotes the fluctuating component. The mean component is then given by:

$$\bar{\phi}(x,y,z,t) = \frac{1}{T}\int_{-T/2}^{T/2} \phi(x,y,z,t+\tau)\mathrm{d}\tau \tag{2.7}$$

where T is the chosen time period for the averaging. Using this approach, the Navier-Stokes equations become the Reynolds Averaged Navier-Stokes (RANS) equations for incompressible flow. For compressible flow this simple approach would require information about density-velocity correlations, hence Favre or mass averaging is instead used (often referred to as Favre Averaged Navier-Stokes (FANS)) for the time-averaged velocities, temperature and total energy [103]. Favre averaging uses simple averaging for density and pressure, but density weighted averages for velocities, temperature and total energy. Using density weighting for a general turbulent variable ϕ gives:

$$\phi = \tilde{\phi} + \phi'' \tag{2.8}$$

where the overbrace denotes the density weighted average component, and the second term is the fluctuating component. The density weighted mean component is then given by:

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}} \tag{2.9}$$

where, like before, the overbar accent denotes a simple time average value. It should therefore be noted that $\overline{\phi''} \neq 0$ but the Favre average is, such that:

$$\tilde{\phi''} = \frac{\overline{\rho\phi''}}{\bar{\rho}} = 0 \tag{2.10}$$

Both the RANS and FANS equations contain additional terms not present in the un-averaged Navier-Stokes equations. These terms are obtained using turbulence models.

2.2.4. Turbulence Modelling

There are a wide range of turbulence models available, which are typically split into categories based on the number of additional transport equations which must be solved; either none, one or two. There is no perfect model, with each having its own strengths and weaknesses which depend on the problem.

No equation models (also known as algebraic models), such as the Baldwin-Lomax [104] and Cebeci-Smith [105] models, calculate the turbulence directly from the flow

variables; rather than solving transport equations. This has advantages in that it makes the models robust and good predictions have been obtained for cases with attached boundary layers under weakly favourable or adverse pressure gradients. However, their drawback is that they are unable to model time related effects such as diffusion or convection and fail to accurately predict properties in separated flows. As a result, they are not widely used, particularly with regards to aerospace industrial applications.

A popular family of one-equation turbulence models are the Spalart-Allmaras models; originally developed by Spalart and Allmaras for aerodynamic flows in 1992 and continuously developed since then [106–110]. This model, and its derivatives, has been used extensively because it gives superior predictions compared to algebraic models for separated flow and includes history effects. It is also robust compared to two-equation models, working even when mesh quality is not optimum in the near wall region.

Two related and traditionally very popular two-equation turbulence models are k-ϵ [111, 112] and k-ω [113, 114]; where here k stands for turbulence kinetic energy, ϵ stands for the rate of turbulence kinetic energy dissipation and ω stands for the specific rate of kinetic energy dissipation. The two models work well together, with the former being poor for flows with strong adverse pressures, jet flow and highly curved flow, and the latter being good for such flows. k-ϵ is perhaps the more popular of the two, due to its extremely good convergence characteristics that arise from being only weakly non-linear. k-ω sacrifices some of this strong convergence in order to be more non-linear; giving it its strength in the aforementioned flow types. Due to the close relationship between these two models, and their complimentary nature to one another, they are combined in a hybrid method called the Shear Stress Transport (SST) model [115, 116]; using k-ϵ for the freestream and k-ω elsewhere. SST provides a model that has the robustness of the k-ϵ model and the improved accuracy of the k-ω model, and as a result is a very common choice of turbulence model.

For the work carried out on this thesis the choice of turbulence model is not a significant factor as the ROMs work on a “what you put in, you get out” basis; that is to say, a ROM built using CFD simulations using a particular turbulence model, will give outputs equivalent to CFD simulations using that same turbulence model. Therefore, a simple to use and robust turbulence model was more desirable than one that improved the accuracy of the CFD slightly. Thus, one of the versions of the Spalart-Allmaras

One-Equation Model (SA) has been selected. The original Spalart-Allmaras model is described first, followed by the various variants that have arisen since it was first described. This calculates the turbulent eddy viscosity (ν_t) as:

$$\nu_t = \bar{\bar{\nu}} f_{v1} \tag{2.11}$$

where the double bar accent denotes a turbulence working variable and $\boldsymbol{f_{v1}}$ is a function defined as:

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \tag{2.12}$$

where c_{v1}^3 is a constant (see Table 2) and χ is a ratio of the turbulence working variable to the laminar kinematic viscosity, such that:

$$\chi = \frac{\bar{\bar{\nu}}}{\nu} \tag{2.13}$$

To solve Eqn. 2.11, a transport equation must be solved to obtain $\bar{\bar{\nu}}$:

$$\begin{aligned} \frac{\partial \bar{\bar{\nu}}}{\partial t} + u_i \frac{\partial \bar{\bar{\nu}}}{\partial x_i} = c_{b1}(1 - f_{t2})\bar{\bar{Z}}\bar{\bar{\nu}} - \left[c_{w1} f_w - \frac{c_{b1}}{\mathrm{K}^2}\right]\left(\frac{\bar{\bar{\nu}}}{d}\right)^2 \\ + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_i}\left((\bar{\bar{\nu}} + \nu)\frac{\partial \bar{\bar{\nu}}}{\partial x_i}\right) + c_{b2}\frac{\partial \bar{\bar{\nu}}}{\partial x_j}\frac{\partial \bar{\bar{\nu}}}{\partial x_j}\right] \end{aligned} \tag{2.14}$$

where K is the von Karman constant; d is the distance to the nearest wall and the following functions are required:

$$f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{\frac{1}{6}} \tag{2.15}$$

$$f_{t2} = c_{t3} \exp(-c_{t4}\chi^2) \tag{2.16}$$

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \tag{2.17}$$

$$g = r + c_{w2}(r^6 - r) \tag{2.18}$$

$$r = \min\left[\frac{\bar{\bar{\nu}}}{\bar{\bar{Z}} K^2 d^2}, 10\right] \tag{2.19}$$

$$\bar{\bar{Z}} = M + \frac{\bar{\bar{\nu}}}{K^2 d^2} f_{v2} \tag{2.20}$$

$$\chi = \frac{\bar{\bar{\nu}}}{\nu} \tag{2.21}$$

where M is the magnitude of the turbulence vorticity, and the constants for the standard version are given in Table 2 below, with the exception of c_{w1} which is relative to other constants such that:

$$c_{w1} = \frac{c_{b1}}{K^2} - \frac{1 + c_{b2}}{\sigma} \tag{2.22}$$

Table 2. Spalart-Allmaras Turbulence Model Constants.

Constant	c_{b1}	c_{b2}	c_{w2}	c_{w3}	c_{v1}	c_{t3}	c_{t4}	σ	K
Value	0.1355	0.622	0.3	2	7.1	1.2	0.5	$0.\dot{6}$	0.41

It should be noted that the value for c_{t3} and c_{t4} were originally 1.1 and 2.0 respectively, but were later updated by the original authors [107].

In 1994 the original authors published a modified version of the SA turbulence model, with a trip term; a method known as Spalart-Allmaras One-Equation Model with Trip Term (SA-la) [107]. The SA-la model is identical to SA except for Equation 2.14 which is replaced with:

$$\begin{aligned}\frac{\partial \bar{\bar{\nu}}}{\partial t} + u_i \frac{\partial \bar{\bar{\nu}}}{\partial x_i} = c_{b1}(1 - f_{t2})\bar{\bar{Z}}\bar{\bar{\nu}} - \left[c_{w1} f_w - \frac{c_{b1}}{K^2}\right]\left(\frac{\bar{\bar{\nu}}}{d}\right)^2 \\ + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_i}\left((\bar{\bar{\nu}} + \nu)\frac{\partial \bar{\bar{\nu}}}{\partial x_i}\right) + c_{b2}\frac{\partial \bar{\bar{\nu}}}{\partial x_j}\frac{\partial \bar{\bar{\nu}}}{\partial x_j}\right] + f_{t1}\Delta u^2\end{aligned} \tag{2.23}$$

Where Δu is the change in velocity between the field point and the trip, and where:

$$f_{t1} = c_{t1} g_{trip} \exp[-c_{t2}\frac{\omega_{trip}}{\Delta u^2}(d^2 + g_{trip}^2 d_{trip}^2) \tag{2.24}$$

and:

$$g_{trip} = \min\left[0.1, \frac{\Delta u}{\omega_{trip}\Delta x_{trip}}\right] \tag{2.25}$$

where Δx_{trip} is the grid spacing on the wall at the trip, d_{trip} is the distance to that wall and ω_{trip} is the wall vorticity at the trip. The constants c_{t1} and c_{t2} are set at 1 and 2 respectively.

In 2012 the original authors published another modified version of the SA turbulence model. Known as the negative Spalart-Allmaras One-Equation Model (SA-neg) [110], this model is based upon the SA model, with the only change being that for negative values of $\bar{\bar{\nu}}$, Equation 2.14 is replaced with:

$$\frac{\partial \bar{\tilde{\nu}}}{\partial t} + u_i \frac{\partial \bar{\tilde{\nu}}}{\partial x_i} = c_{b1}(1 - f_{t2})\bar{\tilde{Z}}\bar{\tilde{\nu}} - c_{w1}\left(\frac{\bar{\tilde{\nu}}}{d}\right)^2 + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_i}\left((\bar{\tilde{\nu}} + \nu f_n)\frac{\partial \bar{\tilde{\nu}}}{\partial x_i}\right) + c_{b2}\frac{\partial \bar{\tilde{\nu}}}{\partial x_j}\frac{\partial \bar{\tilde{\nu}}}{\partial x_j}\right] \tag{2.26}$$

where:

$$f_n = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3} \tag{2.27}$$

for which c_{n1} is a new constant, with the value of 16.

Another variation of the SA model, and the turbulence model used within this thesis, is a modification that simply sets the constant c_{t3} as being equal to 0 (rather than 1.2 as laid out in Table 2), which causes the f_{t2} function to also go to zero [117, 118]. As a result this method is known as the Spalart-Allmaras One-Equation Model without f_{t2} Term (SA-noft2). The purpose of this modification is simply that the f_{t2} was a numerical fix to the standard SA model with regards to the stability caused by a trip; therefore, if no trip is used then the term is redundant and removing it will have negligible difference on the results [119].

2.3. Three-Dimensional Differential Euler Equations

The Navier-Stokes equations include the effects of viscosity, but for many aerospace flows viscous effects are negligible for most of the flow field. It is therefore common to solve the simplified set of equations known as the Euler equations, which can be derived from the Navier-Stokes equation assuming adiabatic, inviscid flow. The Euler equations in terms of dimensional variables are given by:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0 \tag{2.28a}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2 + P}{\partial x} + \frac{\partial \rho uv}{\partial y} + \frac{\partial \rho uw}{\partial z} = 0$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial \rho vu}{\partial x} + \frac{\partial \rho v^2 + P}{\partial y} + \frac{\partial \rho vw}{\partial z} = 0 \tag{2.28b}$$

$$\frac{\partial \rho w}{\partial t} + \frac{\partial \rho wu}{\partial x} + \frac{\partial \rho wv}{\partial y} + \frac{\partial \rho w^2 + P}{\partial z} = 0$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial u(\rho E + P)}{\partial x} + \frac{\partial v(\rho E + P)}{\partial y} + \frac{\partial w(\rho E + P)}{\partial z} = 0 \tag{2.28c}$$

As before pressure is given by:

$$P = (\gamma - 1)\left[\rho E - \frac{\rho(u^2 + v^2 + w^2)}{2}\right] \tag{2.29}$$

2.4. Overview of CFD Solvers

The CFD methodology used for this study is essentially an arbitrary choice as the reduced order modelling is generic, just requiring simulation output from a CFD code. The ROMs produced will, at best recover the CFD output on which it was based including the specific parameter settings used within the code. If the CFD is changed, then the ROM build process will not change, but the ROM output would then be linked to the new CFD output. Hence the CFD is not a major element of this work and so only a brief overview of the main features of CFD solvers is presented, together with more detail on the DLR TAU code used in this work.

2.4.1. Meshes

To carry out a CFD simulation it is generally necessary to produce a mesh (otherwise known as a grid) of the fluid volume surrounding the geometry being investigated. The fluid volume is discretised into a number of cells (also known as elements) and can be either structured or unstructured. Note that no meshing was carried out as part of the work done for this thesis; with meshes either obtained from the industrial sponsors of the PhD or from in house sources.

In two-dimensional meshes, cells are typically either triangles or quadrilaterals, consisting of one-dimensional faces joined together by nodes, see for example Figure 2.1 where the black lines are the faces and the red circles are the nodes. In three-dimensional meshes, cells are typically either tetrahedrons or hexahedrons and consist of two-dimensional faces (which like two-dimensional cells; the only difference being the boundaries are called edges) joined together by nodes; this is demonstrated in Figure 2.2 where the black lines are the edges, the blue square highlights one of the faces, and the red spheres representing the nodes.

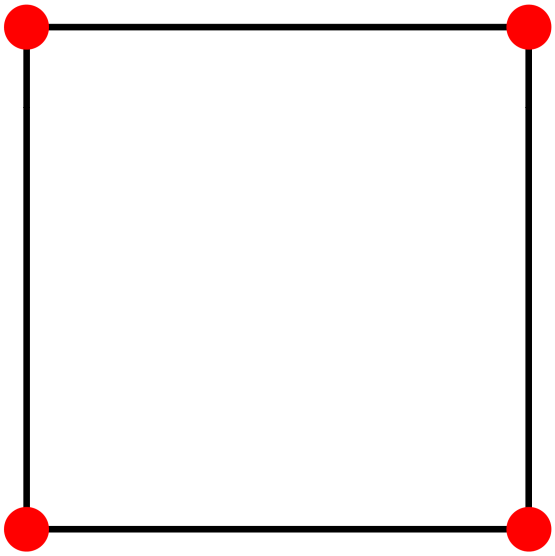

Figure 2.1. Example of a single two-dimensional (quadrilateral) cell.

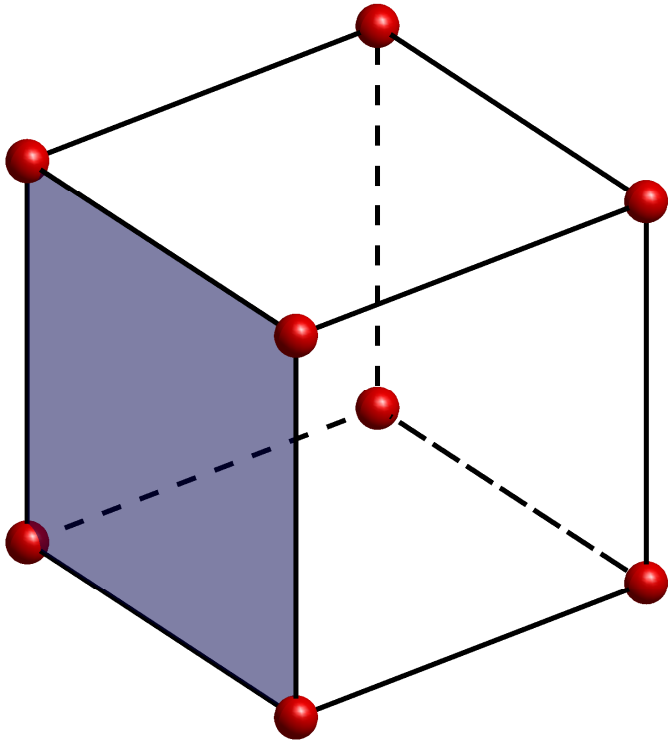

Figure 2.2. Example of a single three-dimensional (hexahedron) cell.

Regardless of the number of dimensions, the mesh can be discretised in such a way that creates either structured or unstructured arrangement of the cells. Structured meshes are characterised by having cells organised in a countable pattern. Conversely, for

unstructured meshes cells not in an organised pattern, and are instead placed and shaped as needed. These two types of mesh are demonstrated in Figures 2.3-2.5 which show a mock-up of a simple two-dimensional geometry, a structured mesh and an unstructured mesh respectively.

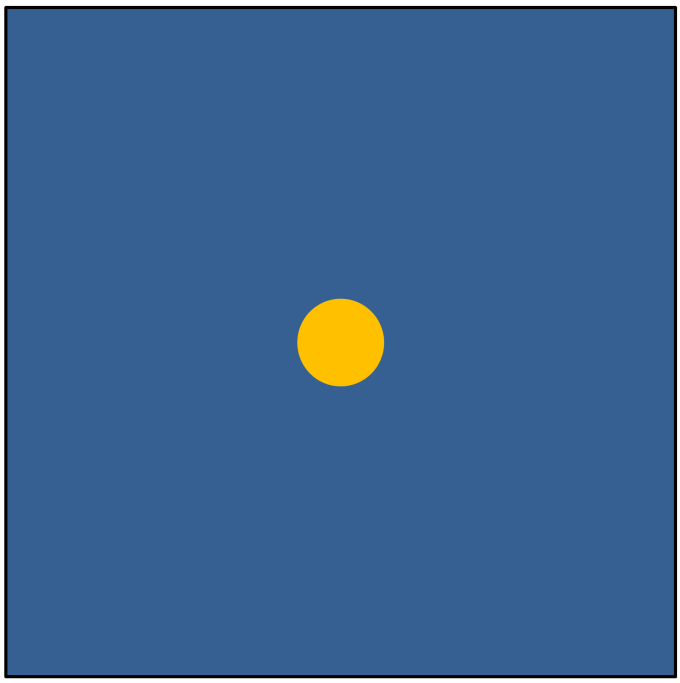

Figure 2.3. Mock-up of a two-dimensional disc geometry (gold) with fluid (blue) contained in a square domain.

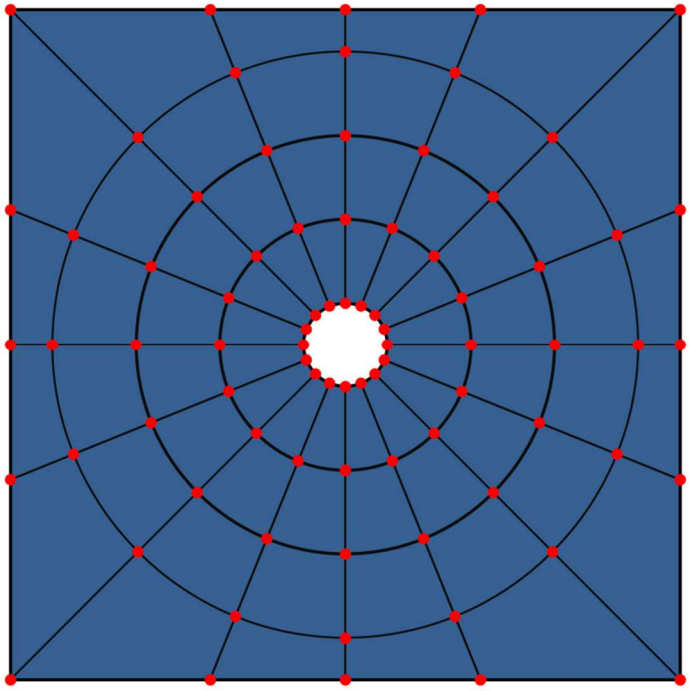

Figure 2.4. Mock-up of a two-dimensional, structured mesh for fluid around a disc geometry within a square domain.

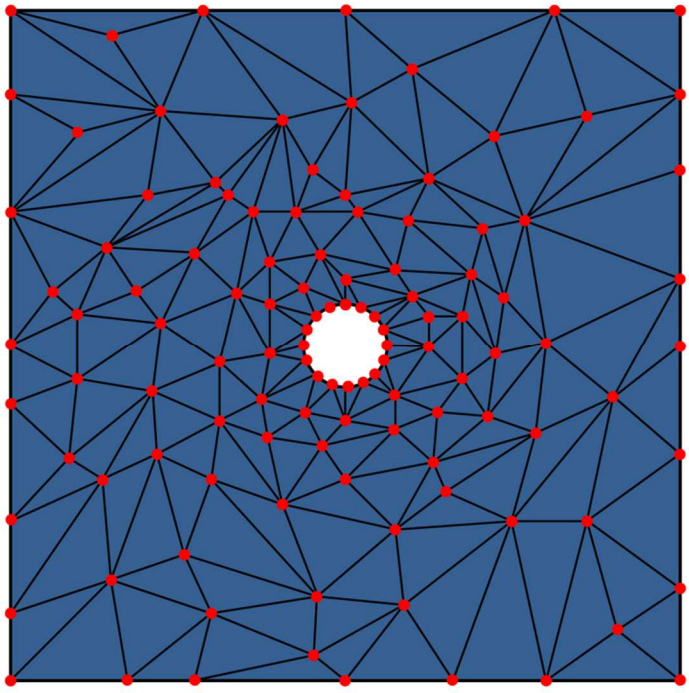

Figure 2.5. Mock-up of a two-dimensional, unstructured mesh for fluid around a disc geometry within a square domain.

Structured meshes have the advantage of being computationally efficient from a memory standpoint as identifying how each cell connects to its neighbours is

straightforward and for simple geometries are quick to generate and can give good boundary layer resolution. Conversely, unstructured meshes require more information to be stored about this cell connectivity, as the nodes are located at effectively random locations. However, whilst unstructured meshes are less memory efficient, they are more flexible in terms of the complexity of geometry they are able to represent and generating meshes for complex aircraft geometries is simpler using automated mesh generators. For viscous flows the cells near the body can become highly skewed which can be problematic [120]. A third type of mesh, known as a hybrid mesh, attempts to blend unstructured and structured meshes to find a reasonable compromise between the memory efficiency of structured meshes, and the flexibility of unstructured meshes.

2.4.2. Spatial Discretisation

There are three main types of spatial discretisation used in CFD solvers; finite difference, finite element and finite volume.

Finite difference methods use Taylor series expansions to approximate derivative terms in the differential form of the governing equations so that algebraic equations are obtained for each node of the mesh. It is particularly suited to simple uniform meshes, but can also be used for non-uniform or curvilinear meshes which are structured; however a transformation between computational and physical space is required with the associated computational overhead rising as the geometry complexity increases. Examples of the simplest finite difference approximations of a first order derivative on a uniformly spaced mesh are given in Eqns 2.30-2.32. These show forward, backward and central difference approximations respectively [121]

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x)}{h} \tag{2.30}$$

$$\frac{df}{dx} \approx \frac{f(x) - f(x-h)}{h} \tag{2.31}$$

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x-h)}{2h} \tag{2.32}$$

where h is the mesh spacing. The forward (Eqn. 2.30) and backward (Eqn. 2.31) approximations are first order accurate and the central (Eqn. 2.32) approximation is second order accurate.

Finite element methods [122, 123], which were originally designed for computational structural mechanics (CSM). The method defines a parametric representation of the flow variables based on shape functions (also called basis functions, trial functions or interpolation functions) for each cell [124]. An integral formulation is then obtained using the method of weighted residuals by multiplying the base equations by a test function before then integrating this over a cell or element [124]. Different methods are obtained depending on the test or weighting functions used. One widely used method is Galerkin method, where the interpolating functions are used as weighting functions. The finite element method can easily be applied on both structured and unstructured meshes, but finite element method is not inherently conservative and so can suffer from stability issues when discontinuities in the flow occur (such as shocks) [125].

The finite volume method [126, 127] uses integral versions of the governing equations, which are then applied to each volume contained in a cell of the discretised mesh. Flux leaving a cell will enter an adjoining cell, thus the scheme is conservative in nature and as such the conservation laws are not only conserved at a domain level, but also at a cellular level. Finite volume is conceptual simple, works well for both structured and unstructured meshes and is easy to implement. This has led to it becoming the most common spatial discretisation scheme used within aerospace based CFD and it is used in the code used to produce the results within this thesis.

2.4.3. Time Stepping

Simulations of compressible flows are usually performed by advancing in time from an initial starting solution, an approach called time-stepping. For steady flows the starting solution is typically free stream conditions (modified near boundaries) or a known solution with different, but similar parameter values. Advancement in time then occurs until the solution converges and no longer changes with time. In this steady flow the time is essentially fictitious. For unsteady flows, simulations start from a steady state solution and advance in time to find the solution at subsequent times. The time-stepping methods used can be either explicit or implicit.

Explicit schemes use only the current solution to calculate the solution at the next time step, whereas implicit schemes use both the current and future solution to calculate the

next time step solution. For both explicit and implicit there are countless methods of execution, all with their own advantages and disadvantages but there are some common issues. Explicit schemes are simple to code and have a relatively low memory requirement but have time-step size restrictions for stability related to the fastest wave speed. This problem can be particularly severe for viscous flows. Further, whilst for steady flows a number of convergence acceleration techniques can be used e.g. local time stepping, these cannot be used for unsteady flows since they destroy time accuracy. Implicit schemes do not have the time step size restrictions for stability, but a matrix equation has to be formed and solved at each time step which imposes a memory requirement that can become very high as the problem size increases. An alternative approach is the dual-time stepping scheme [128–130]. This approach allows an implicit discretisation in real time, but at each time step marches the solution in a fictitious pseudo time using explicit time stepping. Since at each pseudo time step real time is fixed, convergence acceleration such as local pseudo time stepping can be used without compromising time accuracy. Since it is an implicit method, but avoids the solution of a matrix equation it has lower memory requirements than other implicit methods and has thus become a popular method for unsteady flow simulations in aerospace and is used in the CFD code used here.

2.4.4. Non-dimensionalisation

The governing equations are often put into a non-dimensional form so that the magnitudes of variables are normalised (potentially improving accuracy) and dimensionless parameters (or similarity parameters) arise that indicate the relative importance of certain physical flow attributes. Flows over identically shaped, but differently sized geometries will be identical if the similarity parameters are the same and matching these is important when comparing simulations and the results of scale model testing.

Many CFD solvers use non-dimensional variables internally and then also return non-dimensional outputs. However, the CFD solver used in this thesis (DLR-TAU) can have dimensional or non-dimensional inputs, uses non-dimensional variables internally and returns the outputs in dimensional form for dimensional input. The only exception is that non-dimensionalised pressures forces and moments are also returned and these are

the outputs chosen for the ROMs in this thesis, so the ROM output is always non-dimensional. However the ROM input for gusts could either have a dimensional or non-dimensional input gust velocity. There are a number of different possible non-dimensional scalings that could be used. The one used internally by TAU, is given by:

$$t_{non\_dim} = \mathrm{t}\frac{\sqrt{P_\infty/\rho_\infty}}{L} \tag{2.33a}$$

$$P_{non\_dim} = \frac{P}{P_\infty} \tag{2.33b}$$

$$\rho_{non\_dim} = \frac{\rho}{\rho_\infty} \tag{2.33c}$$

$$T_{non\_dim} = \frac{T}{T_\infty} \tag{2.33d}$$

$$\mu_{non\_dim} = \frac{\mu}{\mu_\infty} \tag{2.33e}$$

$$\left(x_{non\_dim}, y_{non\_dim}, z_{non\_dim}\right) = \frac{(x, y, z)}{L} \tag{2.33f}$$

$$\left(u_{non\_dim}, v_{non\_dim}, w_{non\_dim}\right) = \frac{(u, v, w)}{\sqrt{P_\infty/\rho_\infty}} \tag{2.33g}$$

where here T represents temperature, L is the characteristic length scale and subscript non_dim represent the non-dimensionalised quantity and subscript ∞ represents the far field conditions (i.e. the conditions at infinity).

2.4.5. Aeroelastic Modelling

Aeroelastic effects are those induced in a flexible structure by a combination of the aerodynamic, inertia and elastic forces; as shown by Collar's aeroelastic triangle [131] in Figure 2.6.

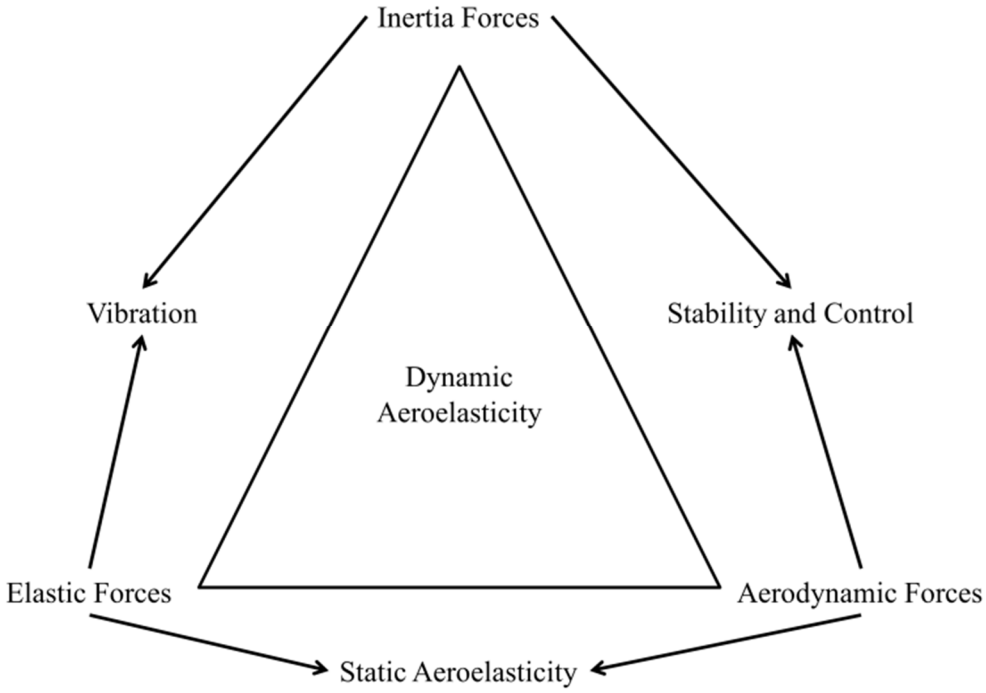

Figure 2.6. Collar's aeroelastic triangle.

As shown in the diagram, aeroelasticity can be loosely split into one of two types; static and dynamic. Similar to steady and unsteady aerodynamics, these two types of aeroelasticity can be defined as instances of aeroelastic deformation which change in time (dynamic) and those which do not (static). As static aeroelasticity is not time dependent, any terms that are dependent on time (such as acceleration, etc.) can be neglected and only steady aerodynamics are required; this ultimately makes static aeroelastic cases less computationally expensive [1].

Being able to accurately model the effects of aeroelasticity is highly important in the design of aircraft, as its effects can have serious ramifications on performance and safety. The effects of static aeroelasticity can include increases in drag (leading to range reduction) as the structure deforms to a sub-optimal shape, reduction (or even reversal) in the efficiency of control surfaces, and divergence, which can lead to structural failure [1]. Dynamic aeroelasticity includes effects such as buffeting and flutter; which again can lead to structural failure [1]. Additionally, it should be noted that improper modelling of the effects of aeroelasticity can make a large difference to the accuracy of the results. This was highlighted by Heinrich *et al* [47] who demonstrated that the root bending moment was significantly affected by the difference structural shapes caused by aeroelastic deformation.

To solve aeroelastic problems, there are two main methodologies. The first method and arguably the most common approach, is to partition the problem into two distinct solvers. A CFD solver to calculate the flow and a CSM solver to calculate the structural deformation. These two solvers are then coupled together, using an interpolating function to map the forces from one solver to another (which often have drastically different grid densities, etc.). Examples of such interpolation functions include fixed-point [132–134], coarse grid predictors [135], approximate Block-Newton [136, 137] and modified Newton-Raphson [138, 139]. Here, the CFD solver will resolve the flow around the initial, non-deformed, geometry; this solution is then used to calculate the forces acting on the structure. These forces are then passed to the structural solver which uses them to calculate the resultant deformation. The next step of the process depends on whether strong or weak coupling is being used. Weak coupling assumes that the dynamic behaviour of the two systems occurs at different speeds and so they do not interact strongly. Conversely, strong coupling does not make such an assumption and so the initial geometry is deformed and the process is repeated until convergence is achieved, or a maximum number of iterations are reached. The alternative method is to use a monolithic solver [140–142], which simultaneously solves the fluid and structural elements of the problem within a single solver; and can therefore be considered fully coupled.

2.5. Flow Solver and Key Settings Used

The CFD software used to produce the CFD results shown in this thesis was the DLR-TAU code [100]. The code is modularised and built around a second order, finite volume flow solver which is designed for parallel processing; which for the work carried out in this thesis was spatially discretised using a classic central Jameson scheme [143]. For viscous cases, the three-dimensional, FANS equations were used along with SA-noft2 to model turbulence. Time stepping was carried using a dual time scheme with a Runge-Kutta pseudo-time-stepping method. In all cases, unstructured, three-dimensional meshes were used and for the aeroelastic case a built-in CSM module was used.

3. Reduced Order Model Theory

This chapter presents the mathematical foundations and practical issues behind the ERA Reduced Order Modelling approach that has been implemented and extended in this research. The description starts from the non-linear fluid equations, in the spatially discrete form found in the CFD code (see Chapter 0). These equations can be written as a non-linear multiple-input-multiple-output (MIMO) system of n-th order:

$$\begin{aligned} \dot{\boldsymbol{x}} &= e(\,\boldsymbol{x}(t),\ \boldsymbol{q}(t)\,) \\ \boldsymbol{y} &= f(\,\boldsymbol{x}(t),\ \boldsymbol{q}(t)\,) \end{aligned} \tag{3.1}$$

Where $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ is the state vector, $\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{p}}$ is the output vector, $\boldsymbol{q} \in \mathbb{R}^{\boldsymbol{m}}$ is the input vector, f and e are functions and the dot accent denotes differentiation with respect to time.

The state vector contains the changes (from the mean values) of the physical variables for each cell. The system inputs and outputs are specified by the user for the problem being investigated. In this study, the system input is the gust velocity (which is a function of time), $v_g(t)$, so that the number of inputs is $m = 1$. The system outputs often used in this work are the changes in lift and pitching moment coefficients from the non-linear steady state values of these coefficients. Thus $p = 2$ and the output vector is given by:

$$\boldsymbol{y} = \begin{bmatrix} \widehat{C}_l \\ \widehat{C}_m \end{bmatrix} \tag{3.2}$$

3.1. Approximation for Weakly Non-Linear Systems

A common assumption in the development of ROMs for unsteady flows is that the dynamic behaviour of the system, relative to an initial non-linear steady mean flow, is approximately linear. Previous work of Wales *et al* [3] has shown that making this assumption leads to ROMs that can predict a range of gust encounters. Further, it was found that at higher gust lengths the linear model accuracy was reduced due to a large shock motion, but that these non-linear flows could be predicted by correcting linear ROMs with steady-state data.

3.1.1. Continuous-Time Systems

If the dynamic behaviour can be assumed to be approximately linear, then the non-linear system (3.1) can be replaced by a linear time-continuous state-space system:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{q}(t) \\ \boldsymbol{y}(t) &= \mathbf{C}\boldsymbol{x}(t) + \mathbf{D}\boldsymbol{q}(t) \end{aligned} \tag{3.3}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{p\times n}$ and $\mathbf{D} \in \mathbb{R}^{p\times m}$ are the system matrices.

To solve this differential equation, Eqn. (3.3) is multiplied through by $e^{-\mathbf{A}t}$ and integrated from 0 to t to give [3]:

$$\begin{aligned} \boldsymbol{x}(t) &= \boldsymbol{\Upsilon}(t)\boldsymbol{x}(0) + \int_0^t \boldsymbol{\Upsilon}(t-\tau)\mathbf{B}\boldsymbol{q}(\tau)d\tau \\ \boldsymbol{y}(t) &= \mathbf{C}\boldsymbol{\Upsilon}(t)\boldsymbol{x}(0) + \int_0^t \mathbf{C}\boldsymbol{\Upsilon}(t-\tau)\mathbf{B}\boldsymbol{q}(\tau)d\tau + \mathbf{D}\boldsymbol{q}(t) \end{aligned} \tag{3.4}$$

Where $\boldsymbol{\Upsilon}(t) = e^{\mathbf{A}t}$. When $\boldsymbol{q} = 0$ then $\boldsymbol{y}(t) = \mathbf{C}\boldsymbol{\Upsilon}(t)\boldsymbol{x}(0)$; which is known as the free response. If $\boldsymbol{x}(0) = 0$ then $\boldsymbol{y}(t) = \int_0^t \mathbf{C}\boldsymbol{\Upsilon}(t-\tau)\mathbf{B}\boldsymbol{q}(\tau)d\tau + \mathbf{D}\boldsymbol{q}(t)$, this is the forced response. For the linearised CFD equations considered here $\boldsymbol{x}(0) = 0$, so the total response of the system can be written as:

$$\begin{aligned} \boldsymbol{x}(t) &= \int_0^t \boldsymbol{\Upsilon}(t-\tau)\mathbf{B}\boldsymbol{q}(\tau)d\tau \\ \boldsymbol{y}(t) &= \int_0^t \mathbf{C}\boldsymbol{\Upsilon}(t-\tau)\boldsymbol{B}\boldsymbol{q}(t)d\tau + \mathbf{D}\boldsymbol{q}(t) \\ &= \int_0^t \big(\mathbf{C}\boldsymbol{\Upsilon}(t-\tau)\boldsymbol{B} + \mathbf{D}\delta(t-\tau)\big)\boldsymbol{q}(t)d\tau \end{aligned} \tag{3.5}$$

where δ is the Dirac delta function.

Introducing the continuous-time impulse response matrix $\mathbf{H}(t) = \mathbf{C}\boldsymbol{\Upsilon}(t)\mathbf{B} + \mathbf{D}\delta(t)$, which is formed from the system output to a unit impulse input on each input channel in turn, the system response $\boldsymbol{y}(t)$ can be written as:

$$\boldsymbol{y}(t) = \int_0^t \mathbf{H}(t-\tau)\boldsymbol{q}(\tau)d\tau \tag{3.6}$$

Although in principle ERA ROMs could be built directly in the continuous-time domain using the $\mathbf{H}(t)$ terms to construct a Hankel matrix, in practice the majority of time-

domain CFD codes/solvers are implemented in discrete-time rather than the continuous-time domain. Thus, it is not easy to obtain the continuous-time impulse response matrix and consequently it is more common to build a discrete-time based ROM. A corresponding continuous-time ROM can then in some cases be obtained from the discrete-time ROM [76].

3.1.2. Discrete Systems

To construct a discrete ROM, it is first necessary to rewrite Eqn. (3.3) in a discrete form. Here, a first order implicit scheme is used:

$$\begin{aligned} \frac{\tilde{\boldsymbol{x}}_j - \tilde{\boldsymbol{x}}_{j-1}}{\Delta t} &= \mathbf{A}\tilde{\boldsymbol{x}}_j + \mathbf{B}\tilde{\boldsymbol{q}}_j \\ \tilde{\boldsymbol{y}}_j &= \mathbf{C}\tilde{\boldsymbol{x}}_j + \mathbf{D}\tilde{\boldsymbol{q}}_j \end{aligned} \tag{3.7}$$

where Δt is the discrete time step, the tilde accent denotes the discrete form of a variable, and subscript j denotes the time level $t = j\Delta t$. This equation can be rearranged to obtain:

$$\begin{aligned} \tilde{\boldsymbol{x}}_j &= \tilde{\mathbf{A}}\tilde{\boldsymbol{x}}_{j-1} + \tilde{\mathbf{B}}\tilde{\boldsymbol{q}}_j \\ \tilde{\boldsymbol{y}}_j &= \tilde{\mathbf{C}}\tilde{\boldsymbol{x}}_j + \tilde{\mathbf{D}}\tilde{\boldsymbol{q}}_j \end{aligned} \tag{3.8}$$

The discrete system matrices are then given in terms of the continuous system matrices by:

$$\begin{aligned} \tilde{\mathbf{A}} &= (\mathbf{I} - \mathbf{A}\Delta t)^{-1} \\ \tilde{\mathbf{B}} &= (\mathbf{I} - \mathbf{A}\Delta t)^{-1}\mathbf{B}\Delta t \\ \tilde{\mathbf{C}} &= \mathbf{C} \\ \tilde{\mathbf{D}} &= \mathbf{D} \end{aligned} \tag{3.9}$$

and the continuous system matrices are given in terms of the discrete system matrices by:

$$\begin{aligned} \mathbf{A} &= \frac{\mathbf{I} - \tilde{\mathbf{A}}^{-1}}{\Delta t} \\ \mathbf{B} &= \frac{\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{B}}}{\Delta t} \\ \mathbf{C} &= \tilde{\mathbf{C}} \\ \mathbf{D} &= \tilde{\mathbf{D}} \end{aligned} \tag{3.10}$$

For CFD codes, the system matrix $\mathbf{D}$ is both generally small and known. Note that for an aircraft's motion the $\widetilde{\mathbf{D}}\widetilde{\boldsymbol{q}}_j$ term represents the effect on the outputs of the displacement of the surface acting on the mean pressure and skin friction. It is then possible to define a modified output vector in Eqn. (3.8) with $(\widetilde{\boldsymbol{y}}_j)^m = \mathbf{C}\widetilde{\boldsymbol{x}}_j$, where the superscript m denotes a modified output system.

$$\begin{aligned} \widetilde{\boldsymbol{x}}_j &= \widetilde{\mathbf{A}}\widetilde{\boldsymbol{x}}_{j-1} + \widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_j \\ (\widetilde{\boldsymbol{y}}_j)^m &= \widetilde{\mathbf{C}}\widetilde{\boldsymbol{x}}_j = \widetilde{\boldsymbol{y}}_j - \widetilde{\mathbf{D}}\widetilde{\boldsymbol{q}}_j \end{aligned} \tag{3.11}$$

It should be noted for the case of gust encounters where $\mathbf{D} = \mathbf{0}$, the original and modified output vector are identical.

The solution of Eqn. (3.11) can be found using one-sided z-transforms [144], so that:

$$(\widetilde{\boldsymbol{y}}_j)^m = \sum_{n=0}^{j} \widetilde{\mathbf{H}}(j-n)\widetilde{\boldsymbol{q}}_r \tag{3.12}$$

where n is the rank of the system and $\widetilde{\mathbf{H}}(j)$ is the discrete-time impulse response matrix at time $j\Delta t$.

If the initial system is undisturbed (i.e. if $\widetilde{\boldsymbol{x}}_{-1} = 0$) then the forced response of the system is given by:

$$\begin{aligned} (\widetilde{\boldsymbol{y}}_0)^m &= \widetilde{\mathbf{C}}\widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_0 \\ (\widetilde{\boldsymbol{y}}_1)^m &= \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_0 + \widetilde{\mathbf{C}}\widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_1 \\ (\widetilde{\boldsymbol{y}}_2)^m &= \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}^2\widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_0 + \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_1 + \widetilde{\mathbf{C}}\widetilde{\mathbf{B}}\widetilde{\boldsymbol{q}}_2 \end{aligned} \tag{3.13}$$

This can be written in matrix form as:

$$\widetilde{\boldsymbol{y}}_\beta^m = \begin{bmatrix} \widetilde{\mathbf{H}}_\beta, & \widetilde{\mathbf{H}}_{\beta-1}, & \ldots, & \widetilde{\mathbf{H}}_2, & \widetilde{\mathbf{H}}_1, & \widetilde{\mathbf{H}}_0 \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{q}}_0 \\ \widetilde{\boldsymbol{q}}_1 \\ \vdots \\ \widetilde{\boldsymbol{q}}_{\beta-2} \\ \widetilde{\boldsymbol{q}}_{\beta-1} \\ \widetilde{\boldsymbol{q}}_\beta \end{bmatrix} \tag{3.14}$$

where $\widetilde{\mathbf{H}}_\beta$ is a simplified writing of $\widetilde{\mathbf{H}}(\beta)$ and the following sequence $(\widetilde{\mathbf{H}}_\beta)$ for $\beta \in \mathbb{Z}_{\geq 0}$:

$$\left\{\widetilde{\mathbf{H}}_0, \quad \widetilde{\mathbf{H}}_1, \quad \ldots, \quad \widetilde{\mathbf{H}}_\beta, \quad \ldots\right\} = \left\{\widetilde{\mathbf{C}}\widetilde{\mathbf{B}}, \quad \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}\widetilde{\mathbf{B}}, \quad \ldots, \quad \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}^\beta\widetilde{\mathbf{B}}, \quad \ldots\right\} \tag{3.15}$$

is called the pulse response of the system.

The pulse response is also the Markov sequence of the system and the terms $\widetilde{\mathbf{H}}_0$, $\widetilde{\mathbf{H}}_1$, … are the Markov parameters. The Markov sequence and the system input uniquely determine the forced response of a linear system. Therefore, if two systems have the same Markov sequence, then they will produce the same forced response for a given

input. It is worth noting that a system of rank n has its forced system response exactly defined by $2n + 1$ Markov parameters. The Markov parameters and the input are the only pieces of data necessary to recreate the response of the discrete linear system for any input.

3.2. System Reduction via Eigensystem Realisation Algorithm

The ROMs in the thesis are created using ERA which was originally developed by Juang and Pappa in 1985 [145] building on the work of Kung [82]. The approach has been successfully applied in a number of studies of fluid flow [52, 61, 67, 81, 82]. The ERA method defines the state-space matrices of a linear ROM in terms of the partitioned matrices found from the SVD of a Hankel Matrix of the linear system. The Hankel matrix is usually built using the pulse responses of the system; however it can be built using alternative data such as sharp-edged gust responses, when seeking a gust ROM as described below.

3.2.1. Step-up and Impulse Responses

A sharp-edged gust in continuous time (see Figure 3.1) has an instantaneous change in gust velocity and the discrete-time approximation of this input sharp edged gust will have the form shown in Figure 3.2, where crosses represent the input values at each time step. The corresponding output response (referred to within this thesis as a step-up response) from either the continuous or discrete time system does not show an instantaneous response; a sketch of a typical response for the force coefficients is shown in Figure 3.3.

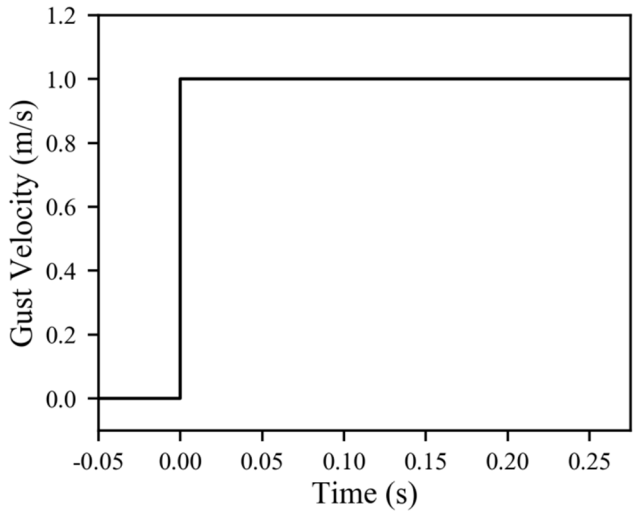

Figure 3.1 Example of the continuous-time input for a sharp-edged gust.

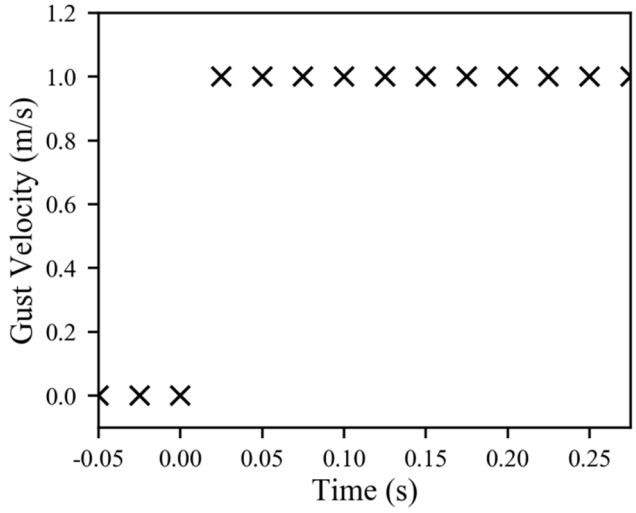

Figure 3.2 Example of the discrete-time input for a sharp-edged gust

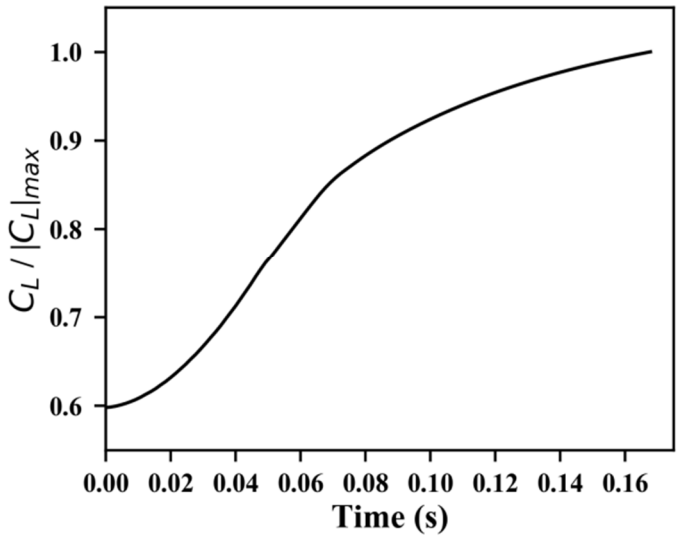

Figure 3.3 Example of step-up response.

A pulse gust input in continuous-time is shown in Figure 3.4, and the equivalent discrete-time input is shown in Figure 3.5. The discrete pulse input can be recreated as a discrete sharp-edge input minus an identical sharp-edge input shifted by one time step, see Figure 3.6, and the discrete step-up input is effectively a summation of pulse responses (see Figure 3.7). Since the system is linear this means that superposition can be used to find a response to the desired input from responses to other inputs. In particular this means that the pulse response of the discrete-time linear system can be found by taking a step-up response, shifting it by one time step and then subtracting it from the original step-up response. A mock-up of this process can be seen in Figures 3.8-3.9.

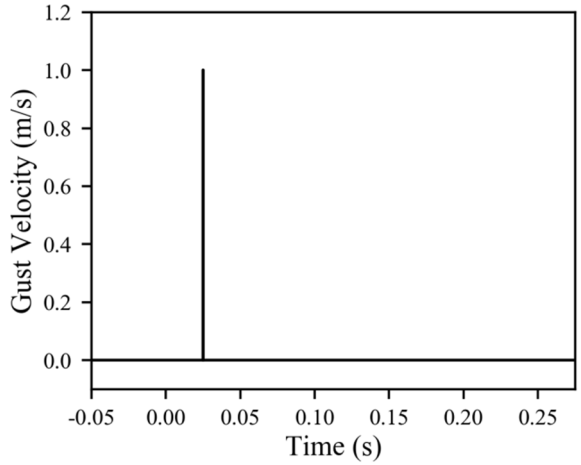

Figure 3.4 Example of the continuous-time input for a pulse gust.

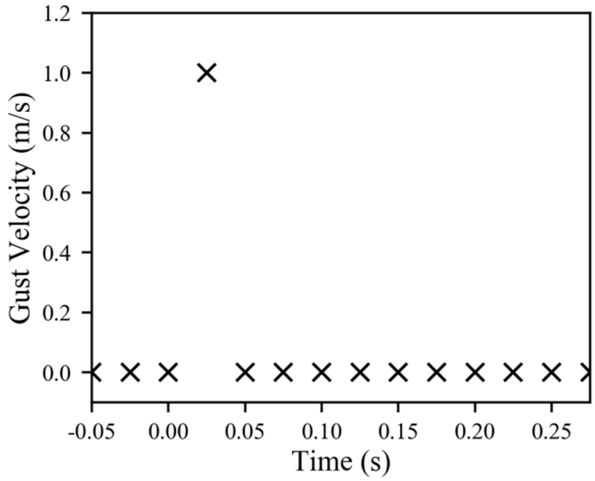

Figure 3.5 Example of the discrete-time input for a pulse gust

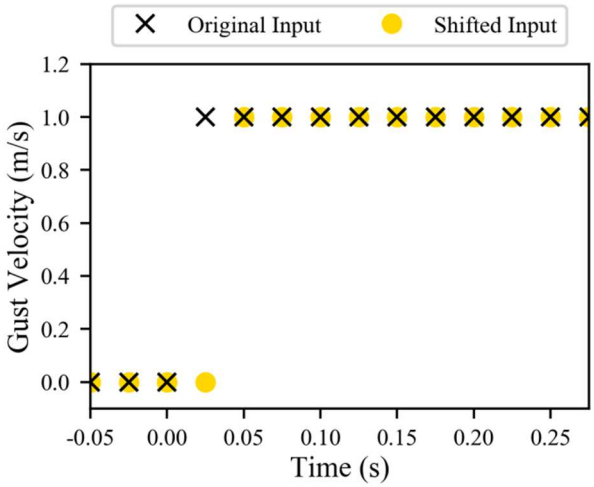

Figure 3.6 Example of the discrete-time input for a pulse gust from two-sharp-edge gusts

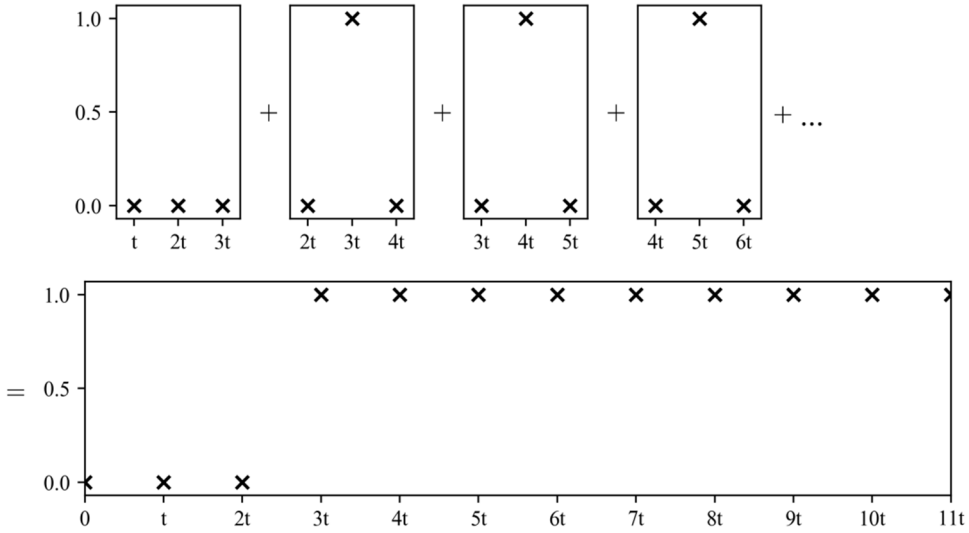

Figure 3.7 Obtaining a discrete step-up input from a set of pulse inputs.

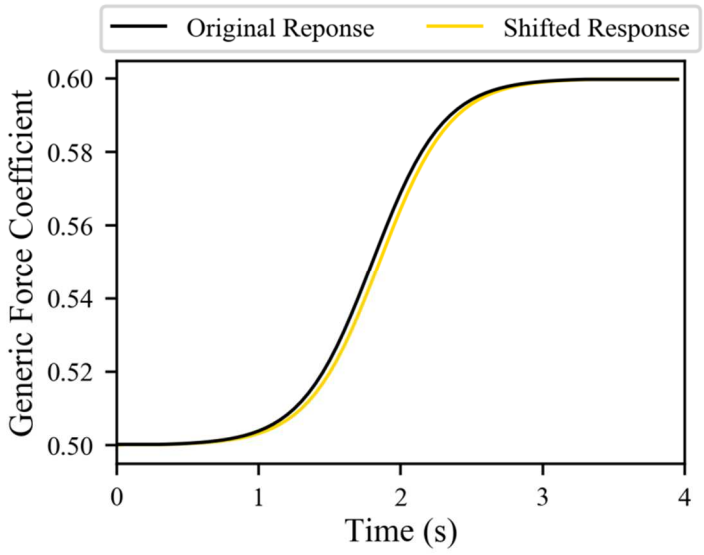

Figure 3.8 Mock-up of the response of a generic force coefficient to a sharp-edged gust, along with a copy that has been shifted by one time step.

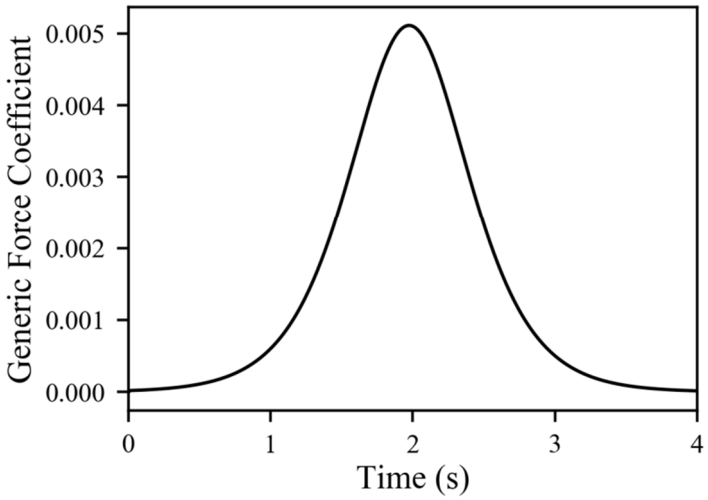

Figure 3.9 Pulse response of a generic force coefficient; created from mock-up of a sharp-edged gust.

3.2.2. Linear Responses from CFD Codes

In order to obtain the linear responses of the CFD code required to create the Hankel matrix for model reduction using ERA, the spatially discrete Euler or Navier-Stokes equations on a mesh can be linearised about a non-linear mean so that the resulting equations have the form of Eqn. (3.11). However, it is not always possible to linearise an existing solver, but Silva [87, 146] showed that the linear response can be approximated as the first order kernel of a Volterra series expansion of the solution. This kernel, $\boldsymbol{\kappa}_1$, captures some level of the amplitude dependence for non-linear systems, and thus in general differs from the purely linear pulse response [89]. However, in cases where the system is weakly-linear for small inputs, then this first order kernel can be assumed to be a good approximation of the response of the linear system. In this case, the linear response can be approximated from two responses of the non-linear CFD code.

As shown by Silva [146], the first order kernel of the system, $\boldsymbol{\kappa}_1$ can be found from:

$$\boldsymbol{\kappa}_1 = 2\boldsymbol{y}_1 - \frac{1}{2}\boldsymbol{y}_{11} \tag{3.16}$$

where $\boldsymbol{y}_{11}$ is the system response to a pulse with a magnitude twice that of $\boldsymbol{y}_1$. Typically these pulse responses are generated via sharp-edged gusts as detailed in Section 3.2.1. The CFD simulations capture the non-linear behaviour of the system, however as long

as the system experiences only near-linear behaviour in response to small inputs, then Eqn. (3.16) allows for the linear behaviour of the system to be approximated.

3.2.3. Reduced Order System Matrices from the SVD of the Hankel Matrix

The ERA method can be used for system identification, where the system matrices of an n^{th}order system can be found using Hankel matrices constructed using $2n + 1$ Markov parameters. However, in the current implementation the system is a large known system resulting from spatial and temporal discretisation of the Euler or Navier-Stokes equations. Thus there is no requirement to identify system matrices since the CFD code already solves the full order system efficiently. Further, the Hankel matrices required would be extremely large and the scheme would not be numerically robust and accurate. In the current application the ERA approach is used for model order reduction, where the dominant behaviour of the system can be found from a relatively small set of impulse response data (training data) and correspondingly small Hankel matrix.

The Hankel matrix is then a $a \times b$ block matrix given by:

$$\mathbf{H}_{ab}(0) = \begin{bmatrix} \mathbf{H}_{\beta} & \mathbf{H}_{\beta+1} & \cdots & \mathbf{H}_{\beta+b-1} \\ \mathbf{H}_{\beta+1} & \mathbf{H}_{\beta+2} & \cdots & \mathbf{H}_{\beta+b} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{\beta+a-1} & \mathbf{H}_{\beta+a} & \cdots & \mathbf{H}_{\beta+a+b-2} \end{bmatrix} \tag{3.17}$$

where in $\mathbf{H}_{\beta}$ each column contains the outputs for a unit pulse input to a single channel at time level β (all other inputs are set to zero). For MIMO systems, with m inputs and p outputs, each Markov parameter is a $p \times m$ matrix. Therefore the size of the Hankel matrix is $ap \times bm$.

The Hankel matrix can then be used to identify reduced system matrices via ERA. The first step of the basic ERA method is the SVD of $\mathbf{H}_{ab}(\beta)$ when $\beta = 0$ which is given by:

$$\mathbf{H}_{ab}(0) = \mathbf{\Omega \Sigma \Theta}^{\mathrm{T}} \tag{3.18}$$

where $\mathbf{\Omega}$ are the left-singular values of $\mathbf{H}_{ab}(0)$, $\mathbf{\Sigma}$ are the singular values of $\mathbf{H}_{ab}(0)$ and $\mathbf{\Theta}$ are the right-singular values of $\mathbf{H}_{ab}(0)$.

$\boldsymbol{\Sigma}$ is a diagonal matrix whose entries, the singular values, are either positive or zero. The singular values in $\boldsymbol{\Sigma}$ are arranged in descending size order. Then a ROM of rank r is found by partitioning the Hankel matrix, with r either defined by the user or found by taking all r singular values of $\boldsymbol{\Sigma}$ which are larger than some desired size. The partitioned Hankel matrix has the following form:

$$\mathbf{H}_{ab}(0) = [\boldsymbol{\Omega}_{red} \quad \boldsymbol{\Omega}_0] \begin{bmatrix} \boldsymbol{\Sigma}_{red} & 0 \\ 0 & \boldsymbol{\Sigma}_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Theta}_{red} \\ \boldsymbol{\Theta}_0 \end{bmatrix}^T \tag{3.19}$$

where subscript red denotes a reduced form, such that $\boldsymbol{\Omega}_{red}$ is of size $ap \times r$, $\boldsymbol{\Sigma}_{red}$ is of size $r \times r$ and $\boldsymbol{\Theta}_{red}$ is of size $bm \times r$.

The Hankel matrix $\mathbf{H}_{ab}(0)$ can then be approximated by:

$$\mathbf{H}_{ab}(0) = \boldsymbol{\Omega}_{red}\boldsymbol{\Sigma}_{red}{\boldsymbol{\Theta}_{red}}^T \tag{3.20}$$

There are several possible realisations of the system and in this work a balanced realisation is used given by [3]:

$$\begin{aligned} \widetilde{\mathbf{A}}_{red} &= \boldsymbol{\Sigma}_{red}^{-\frac{1}{2}}\, \boldsymbol{\Omega}_{red}^T \mathbf{H}_{ab}(1) \boldsymbol{\Theta}_{red} \boldsymbol{\Sigma}_{red}^{-\frac{1}{2}} \\ \widetilde{\mathbf{B}}_{red} &= \boldsymbol{\Sigma}_{\mathrm{red}}^{\frac{1}{2}}\, \mathbf{V}_{red}^T \mathbf{E}_m \\ \widetilde{\mathbf{C}}_{red} &= \mathbf{E}_n^T \mathbf{U}_{red} \boldsymbol{\Sigma}_{red}^{\frac{1}{2}} \end{aligned} \tag{3.21}$$

where $\mathbf{E}_p^T = [\mathbf{I}_p, \ \mathbf{0}_p, \ \mathbf{0}_p, \ \ldots, \ \mathbf{0}_p]$ is a matrix of size $p \times ap$ and $\mathbf{E}_m^T = [\mathbf{I}_m, \ \mathbf{0}_m, \ \mathbf{0}_m, \ \ldots, \ \mathbf{0}_m]$ is a matrix of size $m \times bm$. $\mathbf{I}_p$ and $\mathbf{I}_m$ are the $p \times p$ and $m \times m$ identity matrices respectively.

3.2.4. Alternative ROM Construction Method Using Step-down Responses

It is possible to design an alternative method to construct an ERA based ROM using a step-down response instead of the step-up response based approach described above. By subtracting the step-up input (sharp-edged gust) from the steady gust response, an effective step-down input is produced; shown in continuous space in Figure 3.10. Hence, a step-down response can be found from the step-up response of the force coefficient values. This also applies in discrete space and the step-down response can then be used instead to calculate the system matrices using a slightly modified procedure.

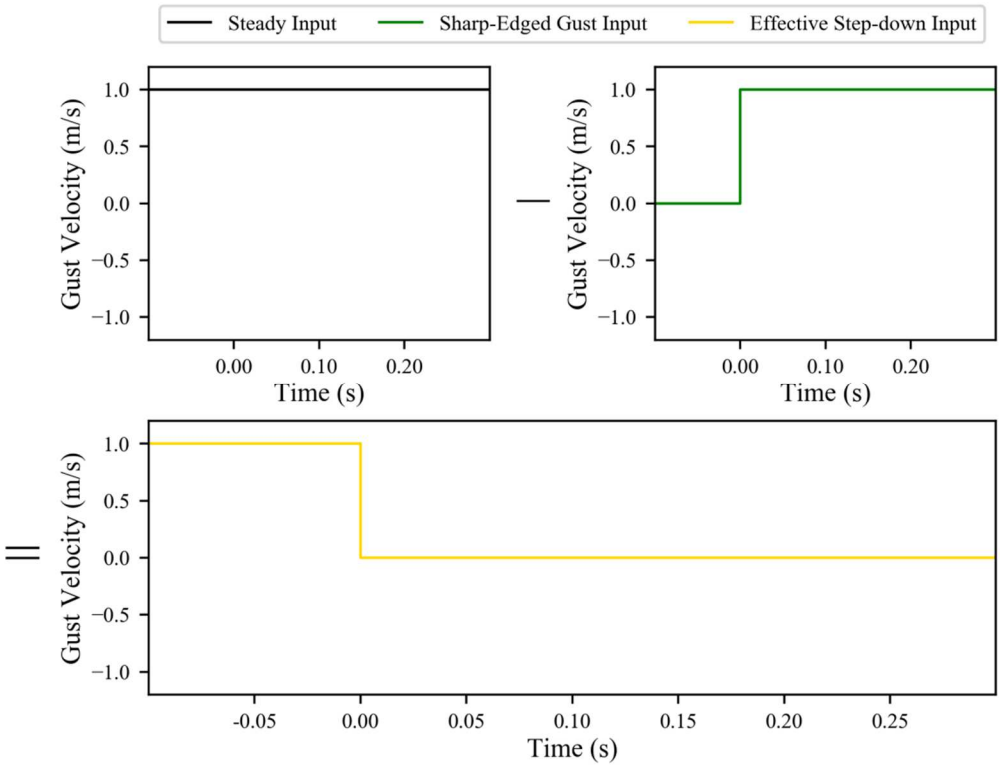

Figure 3.10. Effective step-down input.

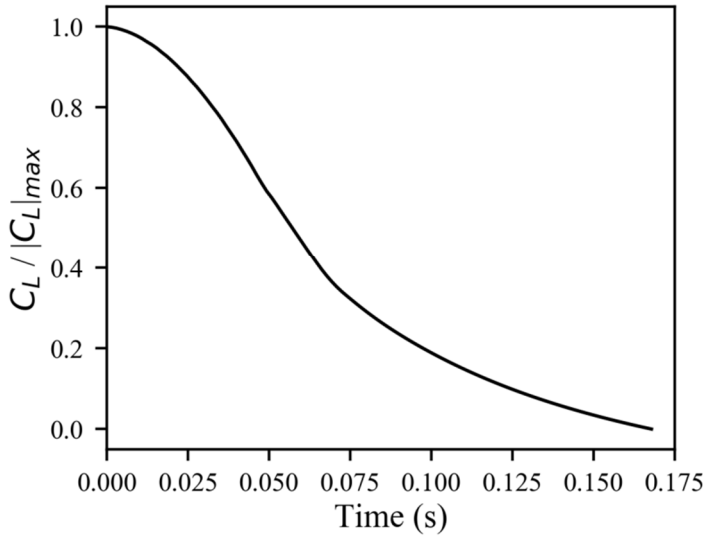

Figure 3.11. Example of step-down response.

Looking at Eqn. (3.3) for a steady state, where $\dot{x} = 0$, gives:

$$0 = \mathbf{A}\boldsymbol{x}(0) + \mathbf{B}\bar{\boldsymbol{q}}(0) \tag{3.22}$$

Where the overbar on q denotes the initial steady state value. This can be rearranged to obtain:

$$\boldsymbol{x}(0) = -\mathbf{A}^{-1}\mathbf{B}\bar{\boldsymbol{q}}(0) \tag{3.23}$$

Since the continuous and discrete systems are identical for an initially steady solution, then for a step-down based system (denoted by the under-bar accent):

$$\underline{\tilde{\boldsymbol{x}}}_0 = -\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}}(0) \tag{3.24}$$

where $\underline{\tilde{\boldsymbol{x}}}_0 = \boldsymbol{x}(0)$ and $\underline{\overline{\boldsymbol{q}}}(0) = \overline{\boldsymbol{q}}(0)$.

Then, using Eqns. (3.8) and setting the input at the first time step $\underline{\tilde{\boldsymbol{q}}}_1 = \underline{0}$ then:

$$\underline{\tilde{\boldsymbol{x}}}_1 = \tilde{\mathbf{A}}\,\underline{\tilde{\boldsymbol{x}}}_0 + \tilde{\mathbf{B}}\,\underline{\tilde{\boldsymbol{q}}}_1 \tag{3.25}$$

Then substituting for $\underline{\tilde{\boldsymbol{x}}}_0$ from Eqn. (3.24) gives :

$$\begin{aligned} \underline{\tilde{\boldsymbol{x}}}_1 &= \tilde{\mathbf{A}}\left(-\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}}\right) + \tilde{\mathbf{B}}\underline{0} \\ \underline{\tilde{\boldsymbol{y}}}_1 &= \tilde{\mathbf{C}}\tilde{\mathbf{A}}\left(-\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}}\right) \end{aligned} \tag{3.26}$$

Then:

$$\begin{aligned} \underline{\tilde{\boldsymbol{x}}}_2 &= \tilde{\mathbf{A}}\underline{\tilde{\boldsymbol{x}}}_1 \\ \underline{\tilde{\boldsymbol{y}}}_2 &= \tilde{\mathbf{C}}\underline{\tilde{\boldsymbol{x}}}_1 \end{aligned} \tag{3.27}$$

Hence:

$$\begin{aligned} \underline{\tilde{\boldsymbol{x}}}_2 &= \tilde{\mathbf{A}}^2\left(-\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}}\right) \\ \underline{\tilde{\boldsymbol{y}}}_2 &= \tilde{\mathbf{C}}\tilde{\mathbf{A}}^2\left(-\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}}\right) \end{aligned} \tag{3.28}$$

$\underline{\tilde{\boldsymbol{x}}}_k$ and $\underline{\tilde{\boldsymbol{x}}}_\beta$ take a similar form for all higher values of k. Thus, a Hankel matrix can be constructed with entries (denoted here for the step-down based method by $\boldsymbol{h}$) that can be written as:

$$\begin{aligned} \boldsymbol{h}_0 &= -\tilde{\mathbf{C}}\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}} \\ \boldsymbol{h}_1 &= -\tilde{\mathbf{C}}\tilde{\mathbf{A}}\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}} \\ \boldsymbol{h}_\beta &= -\tilde{\mathbf{C}}\tilde{\mathbf{A}}^\beta\mathbf{A}^{-1}\mathbf{B}\underline{\overline{\boldsymbol{q}}} \end{aligned} \tag{3.29}$$

If we replace $-\mathbf{A}^{-1}\mathbf{B}$ with $\hat{\mathbf{B}}$ (where the hat accent simply denotes a modified $\mathbf{B}$ matrix) then the Hankel matrix becomes:

$$\mathbf{H}(0) = \begin{bmatrix} -\tilde{\mathbf{C}}\hat{\mathbf{B}} & -\tilde{\mathbf{C}}\tilde{\mathbf{A}}\hat{\mathbf{B}} & \cdots & -\tilde{\mathbf{C}}\tilde{\mathbf{A}}^\beta\hat{\mathbf{B}} \\ -\tilde{\mathbf{C}}\tilde{\mathbf{A}}\hat{\mathbf{B}} & -\tilde{\mathbf{C}}\tilde{\mathbf{A}}^2\hat{\mathbf{B}} & & \vdots \\ \vdots & & \ddots & \vdots \\ -\tilde{\mathbf{C}}\tilde{\mathbf{A}}^\beta\hat{\mathbf{B}} & \ldots & \ldots & -\tilde{\mathbf{C}}\tilde{\mathbf{A}}^{2\beta+1}\hat{\mathbf{B}} \end{bmatrix} \tag{3.30}$$

where:

$$\begin{aligned} \tilde{\mathbf{A}} &= (\mathbf{I} - \mathbf{A}\Delta t)^{-1} \\ \hat{\mathbf{B}} &= -\mathbf{A}^{-1}\mathbf{B} \end{aligned} \tag{3.31}$$

Note that $\tilde{\mathbf{A}}$ is the same as in (3.9) and that in this case the ERA method will decompose the Hankel matrix (3.30) and produce reduced order matrices $\tilde{\mathbf{A}}_{red}$, $\hat{\mathbf{B}}_{red}$ and $\tilde{\mathbf{C}}_{red}$. To

get an expression for $\widetilde{\mathbf{B}}_{red}$ the relationship between the various system matrices must be established

Starting by rearranging the equation for $\widehat{\mathbf{B}}$ in Eqn. (3.31) as:

$$\mathbf{B} = -\mathbf{A}\widehat{\mathbf{B}} \tag{3.32}$$

and substituting into the relationship between the discrete system matrix $\widetilde{\mathbf{B}}$ and the continuous system matrices given in Eqn. (3.9) gives:

$$\widetilde{\mathbf{B}} = (\mathbf{I} - \mathbf{A}\Delta t)^{-1}\mathbf{B}\Delta t = -(\mathbf{I} - \mathbf{A}\Delta t)^{-1}\mathbf{A}\widehat{\mathbf{B}}\Delta t \tag{3.33}$$

gives:

$$\widetilde{\mathbf{B}} = -\widetilde{\mathbf{A}}\mathbf{A}\widehat{\mathbf{B}}\Delta t \tag{3.34}$$

Then substituting for $\mathbf{A}$ from Eqn. (3.10):

$$\widetilde{\mathbf{B}} = -\widetilde{\mathbf{A}}\left(\mathbf{I} - \widetilde{\mathbf{A}}^{-1}\right)\frac{1}{\Delta t}\,\widehat{\mathbf{B}}\Delta t = (\mathbf{I} - \widetilde{\mathbf{A}})\widehat{\mathbf{B}} \tag{3.35}$$

Thus the discrete reduced order system matrix $\widetilde{\mathbf{B}}_{red}$ can be found from the reduced matrices output from the ERA step-down method:

$$\widetilde{\mathbf{B}}_{red} = (\mathbf{I} - \widetilde{\mathbf{A}}_{red})\widehat{\mathbf{B}}_{red} \tag{3.36}$$

It is possible to obtain the continuous reduced order system matrices directly from the matrices output from the step-down Hankel ERA method via:

$$\begin{aligned} \mathbf{A}_{red} &= \left(\mathbf{I} - \widetilde{\mathbf{A}}_{red}{}^{-1}\right)\frac{1}{\Delta t} \\ \mathbf{B}_{red} &= -\mathbf{A}_{red}\widehat{\mathbf{B}}_{red} \\ \mathbf{C}_{red} &= \tilde{\mathbf{C}}_{red} \end{aligned} \tag{3.37}$$

3.3. Steady State Correction

In cases where non-linearities in the response are no longer negligible such as due to shock wave motion, previous work [147] has shown that accuracy of the ROM can be improved by introducing a steady state correction. This is implemented by introducing a diagonal scaling matrix $\mathbf{S}$ in the output Eqn. (3.11):

$$\begin{aligned} \widetilde{\boldsymbol{x}}_j &= \mathbf{A}\widetilde{\boldsymbol{x}}_{j-1} + \mathbf{B}\widetilde{\boldsymbol{q}}_j \\ \left(\widetilde{\boldsymbol{y}}_j\right)^m &= \mathbf{S}\widetilde{\boldsymbol{q}}_j\mathbf{C}\widetilde{\boldsymbol{x}}_j \end{aligned} \tag{3.38}$$

The entries in the diagonal scaling matrix, **S**, are calculated for each force coefficient output denoted by subscript i in Eqn. (3.39), by using the steady states for an equivalent change in angle of attack based on the gust velocity and is given by**:**

$$\mathbf{S}_{ii}(\widetilde{\boldsymbol{q}}_j) = \frac{\overline{\boldsymbol{y}}_i(\widetilde{\boldsymbol{q}}_{CFD})_j}{\overline{\boldsymbol{y}}_i(\widetilde{\boldsymbol{q}}_{Linear})_j} \tag{3.39}$$

The steady state solution, $\overline{\boldsymbol{y}}$, for a fixed $\widetilde{\boldsymbol{q}}$ is given by:

$$\overline{\boldsymbol{y}} = \begin{bmatrix} \hat{C}_l \\ \hat{C}_m \end{bmatrix}_{Steady} \tag{3.40}$$

It is also worth noting that $\overline{\boldsymbol{y}}_i(\widetilde{\boldsymbol{q}}_{Linear})_j$ is often referred to as the linear gradients.

3.4. Stabilisation of Reduced Order Models

The above ERA process is not guaranteed to produce a stable ROM. As described in Section 1.5.5, the stability of a system is related to the system matrix, for a continuous reduced order system, the eigenvalues of $\mathbf{A}_{red}$ must have negative real parts. For a discrete system all the eigenvalues of $\widetilde{\mathbf{A}}_{red}$ must have an absolute value less than one; and thus fall within a unit circle (as shown in Fig. 1.2). Should one or more of the eigenvalues fall outside this range, then the system is considered unstable, and will exhibit behaviours such as exponential growth, oscillation, etc.; typically rendering the system unusable for most applications. Thus, ensuring a system is stable is of extreme importance.

3.4.1. Restarting of a Discrete System

To ensure the discrete ROM is stable, a technique known as restarting can be used. This method was originally demonstrated on Arnoldi and Lanczos methods [96, 97] before being applied to ERA explicitly, by Wales *et al* [6].

Having identified that $\widetilde{\mathbf{A}}$ has unstable eigenvalues, the restarting process attempts to remove unstable eigenvalues starting from the largest such eigenvalue. Simply deleting the eigenvalue and keeping all others fixed will destroy ROM accuracy. Instead the unstable eigenvalues are removed using restarting. The first step in restarting is to

define a new $\widetilde{\mathbf{B}}$ matrix (denoted by $\breve{\mathbf{B}}$). If the largest unstable eigenvalue of $\widetilde{\mathbf{A}}$ is real then $\breve{\mathbf{B}}$ is defined by having a shift applied such that:

$$\breve{\mathbf{B}} = \left(\widetilde{\mathbf{A}} - \lambda_{\mathbb{R}}\mathbf{I}_{\mathrm{r}}\right)\widetilde{\mathbf{B}} \tag{3.41}$$

where λ is the eigenvalue and the subscript $\mathbb{R}$ denotes that it is real. Since $\widetilde{\mathbf{A}}$ is real, a complex conjugate pair of eigenvalues may be the largest unstable eigenvalues. In this situation the new matrix $\breve{\mathbf{B}}$ has a pair of shifts defined by the real and imaginary components of one of the eigenvalues is required:

$$\breve{\mathbf{B}} = \left(\widetilde{\mathbf{A}} - \lambda^{*}\mathbf{I}_{\mathrm{r}}\right)\left(\widetilde{\mathbf{A}} - \lambda^{*}\mathbf{I}_{\mathrm{r}}\right)\widetilde{\mathbf{B}} = (\widetilde{\mathbf{A}}^{2} - 2\lambda_{\mathbb{R}}\widetilde{\mathbf{A}} + (\lambda_{\mathbb{R}}^{2} + \lambda_{\mathbb{I}}^{2})\mathbf{I}_{\mathbb{R}})\widetilde{\mathbf{B}} \tag{3.42}$$

where the superscript $*$ represents the complex conjugate and subscripts $\mathbb{R}$ and $\mathbb{I}$ denote the real and imaginary part of one of the complex pair of unstable eigenvalues respectively.

Once the new matrix $\breve{\mathbf{B}}$ has been calculated, an updated Hankel matrix can be formed such that:

$$\breve{\mathbf{H}}_{ab}(j) = \begin{bmatrix} \tilde{\mathbf{C}} \\ \tilde{\mathbf{C}}\widetilde{\mathbf{A}} \\ \vdots \\ \tilde{\mathbf{C}}\widetilde{\mathbf{A}}^{a-1} \end{bmatrix} [\breve{\mathbf{B}}, \quad \widetilde{\mathbf{A}}\breve{\mathbf{B}}, \quad \cdots, \quad \widetilde{\mathbf{A}}^{b-1}\breve{\mathbf{B}}\,] \tag{3.43}$$

This updated Hankel matrix can also be calculated from the original Hankel matrices. For real eigenvalues this can be calculated as:

$$\breve{\mathbf{H}}_{ab}(j) = \mathbf{H}_{\mathrm{ab}}(j+1) - \lambda_{\mathbb{R}}\mathbf{H}_{\mathrm{ab}}(j) \tag{3.44}$$

and for complex eigenvalues, this updated Hankel matrix is given by:

$$\breve{\mathbf{H}}_{ab}(j) = \mathbf{H}_{\mathrm{ab}}(j+2) - (2\lambda_{\mathbb{R}})\mathbf{H}_{\mathrm{ab}}(j+1) + (\lambda_{\mathbb{R}}^{2} + \lambda_{\mathbb{I}}^{2})\mathbf{H}_{\mathrm{ab}}(j) \tag{3.45}$$

During this process, additional Markov parameters (i.e. Markov parameters not used in the original construction of the Hankel matrix) may be used to maintain the size of the Hankel matrix. However, should additional Markov parameters not be available then the Hankel size will be reduced by one when removing an unstable, real eigenvalue or by two when removing an unstable, complex eigenvalue; this can therefore reduce the maximum size of the ROM. Regardless, the updated Hankel matrix is then used to build a new ROM, which is again checked for stability and the process repeated should unstable eigenvalues remain. Once this process has been completed, and the system consists purely of stable eigenvalues, then the final $\widetilde{\mathbf{B}}$ matrix can be obtained from:

$$\widetilde{\mathbf{B}} = \left(\prod_{i=1}^{ns}(\widetilde{\mathbf{A}} - \lambda_{i}\mathbf{I}_{r-ns})^{-1}\right)\breve{\mathbf{B}} \tag{3.46}$$

where subscript ns denotes the total number of shifts applied during restarting.

3.4.2. Stabilisation Using Schur Decomposition

Another method to stabilise a ROM was put forward by McKelvey *et al* [98]. It utilises Schur decomposition to allow the unstable eigenvalues to be projected back within the stable region.

Schur decomposition states that any square matrix can be decomposed such that:

$$\mathbf{M} = \mathbf{QUQ}^{-1} \tag{3.47}$$

where $\mathbf{M}$ is the matrix being decomposed, $\mathbf{Q}$ is a unitary matrix and $\mathbf{U}$ is an upper triangular matrix with the eigenvalues of $\mathbf{M}$ along the main diagonal.

If the eigenvalues of $\widetilde{\mathbf{A}}_{red}$ contain unstable values, then $\widetilde{\mathbf{A}}_{red}$ can be decomposed by Schur decomposition and the eigenvalues modified. Extremely unstable eigenvalues (with an absolute value greater than 2) are set to zero, eigenvalues with an absolute value of exactly 1 have a small value subtracted to move them within the stable region, and all other unstable eigenvalues have the Eqn. (3.48) applied [98]:

$$\lambda_{stable} = \lambda_{unstable}\left(\frac{2}{|\lambda_{unstable}|} - 1\right) \tag{3.48}$$

Once all the eigenvalues are within the discrete stable region, they are used to construct an updated version of the upper triangular matrix, $\mathbf{U}$, which is then used within Eqn. (3.48) to produce a stable $\widetilde{\mathbf{A}}_{red}$ matrix.

3.4.3. Stabilisation of a Continuous Reduced Order Model

Whilst this thesis only presents discrete ROMs, it is worth noting the differences when stabilising discrete and continuous ROMs; with the primary difference being the location of the stable regions for eigenvalues. Discrete systems have the boundary of the stable region located in a circle of radius 1, located around the origin, whereas continuous systems have the boundary of the stable region located to the left of the complex axis. However, when the restarting takes place in the discrete system the stable region depends of the method used to transform the discrete to the continuous system. For example, if the first order implicit discretisation is used then for the continuous

system to be stable the discrete eigenvalues must be contained in a circle of radius 0.5 around the coordinate (0.5, 0.0i); this is shown in Figure 3.12:

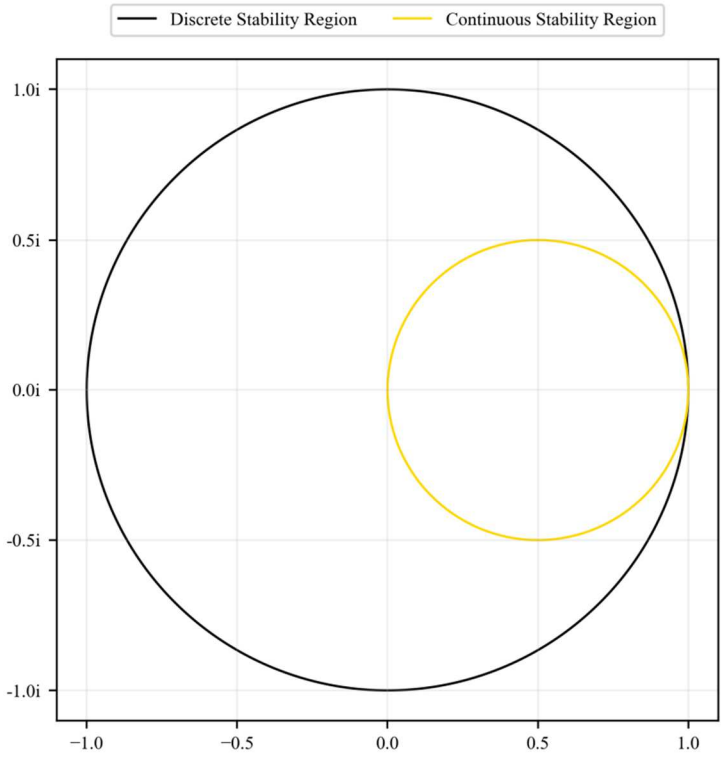

Figure 3.12. Stability regions for eigenvalues of discrete and continuous based ROMs.

As the two stability regions are different, a stable discrete ROM might not map to a stable continuous ROM. For the work carried out in this thesis this isn't a problem as only discrete ROMs are looked at, and no changes to their time step sizes (post gust-impact) are desired. However, it is worth noting that should time step size flexibility be desired, then the differences between discrete and continuous ROMs would have to be accounted for [6].

3.5. ROM Adaptation

3.5.1. Altitude Adaptation

Formally a ROM is only valid for the flight point (Mach and altitude) at which it was originally created. However, advantage can be taken of non-dimensional scaling (see Section 2.4.4) to extend its range of validity to different altitudes than the one used in its creation, so long as the Mach number is kept the same and variations in Reynolds number do not have a significant impact on the flow. If these conditions are met, it is possible to use a single ROM and rescale to obtain a ROM for a range of altitudes. This

results in computational savings as all the ROMs can be obtained from a single set of CFD simulations.

The approach taken to create ROMs at modified altitudes is described below and works on the principle of a non-dimensionalised step-down response. Once this step-down response is rescaled for the new altitude it can be used as the input to the ERA process to create a new ROM. The first stage in achieving this is to run the CFD simulations for an arbitrary altitude (in this thesis sea level has been used to give the greatest possible variation). Next the step-down CFD response (detailed in Section 3.2.4) is modified by rescaling the recorded dimensional time using Eqn. (3.49):

$$t_{new} = t_{old} \frac{\left(\sqrt{p_\infty/\rho_\infty}\right)_{orig_{alt}} \frac{L}{L}}{\left(\sqrt{p_\infty/\rho_\infty}\right)_{new_{alt}}} = t_{old} \frac{(a_\infty)_{orig_{alt}}}{(a_\infty)_{new_{alt}}} = t_{old} \frac{(u_\infty)_{orig_{alt}}}{(u_\infty)_{new_{alt}}} \quad (3.49)$$

where here a represents the speed of sound, subscript new represents an updated value, subscript old represents the old value, subscript $orig_{alt}$ represents a value at the original altitude and subscript new_{alt} represents a value at the new altitude.

As the Mach number, physical lengths used and specific heat ratio are kept fixed, this represents the non-dimensionalisation of the time at the original altitude and the re-dimensionalisation at the new altitude. This modification corrects the rate at which the model responds to the gust by changing the rate at which it effectively passed through the sharp-edged gust used to create the step-down response.

Next the amplitude of the step-down response must be modified. As the outputs are already non-dimensional forces and moments (see Section 2.4.4), only the gust magnitude used to create the response needs to be accounted for. This re-dimensionalisation can be achieved in the same manner as the time using Eqn. (3.49). An equivalent method of rescaling is to use the step-down response with the modified time but unmodified amplitude to create the ROM and then modify the amplitude of the gust input using equation Eqn. (3.50).

$$(v_{g_{ds}})_{new} = (v_{g_{ds}})_{old} \frac{(u_\infty)_{orig_{alt}}}{(u_\infty)_{new_{alt}}} \quad (3.50)$$

Together, these corrections allow for the ROM to be built at one altitude and then used for any altitude so long as the Mach number remains constant; this can drastically reduce the computational cost if multiple altitudes are required per Mach number.

The variation in Reynolds number is an important consideration that is not captured in the process just described. It should be noted that, assuming constant Mach number and standard atmospheric relations, the Reynolds number at an altitude of 10,000m will be approximately one third of the Reynolds number at sea level, hence care would need to be taken in Reynolds number ranges where significant flow changes occur (which is further discussed in Section 8.2).

3.5.2. Surface Pressure Reconstruction

The ROM construction methods presented to this point have only considered the calculation of the time history of force coefficients. However, with modification it is possible to adapt one of these ROMs to also reconstruct the pressures on the surface of the model. This involves building a ROM as normal, and then using the surface pressures of the sharp-edged gust input within a post-process.

It is the view of the author that in aerospace applications the ERA based methods are best suited to calculating component loads or component stresses directly. However, to fit in with existing systems it may be desirable to reconstruct the surface pressures so that a more usual loads process can be followed; hence the interest in pressure reconstruction.

The pressure reconstruction method requires the use of a small selection of the primal modal matrix, which is not something calculated as standard within an ERA based ROM. Therefore, it is necessary to consider ABPOD (see Section 1.5.2). ABPOD has been detailed and compared to the ERA method extensively by Rowley [70] and Ma *et al* [80], who showed that the two methods are equivalent. However, one large advantage of ERA compared to ABPOD is that it does not require adjoint data, nor does it require extremely large matrices to be stored or multiplied. The largest matrices in question are the those for the discrete solutions from an impulse (or step-down) response simulation for the primal system ($\widetilde{\mathbf{X}}$) and those of the adjoint system ($\widetilde{\mathbf{Y}}$), these matrices have one dimension defined by the number of time steps used (although this can be reduced by

only sampling every s time steps) and the other dimension being defined by the size of the CFD solution. Therefore, even for a relatively small simulation such as the wing model used in Chapter 0, each of $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$ can have hundreds of millions of elements. Thus the matrices quickly become unmanageable; making ERA a much more practical ROM construction method.

As the two methods (ABPOD and ERA) are equivalent, we can look to the basic ABPOD method (as shown by Ma *et al* [80]) for a method to calculate the primal modal matrix. This method first involves running an impulse response for the primal system (Eqn. (3.8)) and collecting $\gamma_c + 1$ snapshots of the state vector over $\gamma_c s + 1$ steps (where s is the frequency of sampling; which is 1 in cases where each time step is to be used), such that:

$$\widetilde{\mathbf{X}} = [\widetilde{\mathbf{B}}, \quad \widetilde{\mathbf{A}}^{s}\widetilde{\mathbf{B}}, \quad \widetilde{\boldsymbol{A}}^{2s}\widetilde{\mathbf{B}}, \quad \cdots, \quad \widetilde{\boldsymbol{A}}^{\gamma_c s}\widetilde{\mathbf{B}}\,] \tag{3.51}$$

Then if the adjoint system is considered such that:

$$\tilde{\boldsymbol{z}}_j = \widetilde{\mathbf{A}}^{*}\tilde{\boldsymbol{z}}_{j-1} + \tilde{\mathbf{C}}^{*}\tilde{\boldsymbol{q}}_j \tag{3.52}$$

where the superscript asterisk denotes the adjoint of a matrix (which is also the transpose of a matrix for linear cases). Then, similar to the primal system, $\gamma_o + 1$ snapshots of the adjoint system state vector can be collected, over $\gamma_o s + 1$ steps. Thus:

$$\widetilde{\mathbf{Y}} = [\tilde{\mathbf{C}}^{*}, \quad (\widetilde{\mathbf{A}}^{*})^{s}\tilde{\mathbf{C}}^{*}, \quad (\widetilde{\boldsymbol{A}}^{*})^{2s}\tilde{\mathbf{C}}^{*}, \quad \cdots, \quad (\widetilde{\boldsymbol{A}}^{*})^{\gamma_c s}\tilde{\mathbf{C}}^{*}\,] \tag{3.53}$$

Therefore, the Hankel matrix can be obtained by:

$$\widetilde{\mathbf{H}} = \widetilde{\mathbf{Y}}^{\mathrm{T}}\widetilde{\mathbf{X}} \tag{3.54}$$

After this, the process is similar to the ERA method, with SVD being used to obtain $\boldsymbol{\Omega}_{red}$, $\boldsymbol{\Sigma}_{red}$ and $\boldsymbol{\Theta}_{red}$; as laid out in Eqn. (3.19). From these the primal and adjoint modes of the system are given by:

$$\begin{aligned} \boldsymbol{\Phi}_{\mathrm{red}} &= \widetilde{\mathbf{X}}\boldsymbol{\Theta}_{red}\boldsymbol{\Sigma}_{red}^{\ -\frac{1}{2}} \\ \boldsymbol{\Psi}_{\mathrm{red}} &= \widetilde{\mathbf{Y}}\boldsymbol{\Omega}_{red}\boldsymbol{\Sigma}_{red}^{\ -\frac{1}{2}} \end{aligned} \tag{3.55}$$

where $\boldsymbol{\Phi}_{\mathrm{red}}$ is the reduced primal modal matrix formed from columns of the reduced primal modes of the system and $\boldsymbol{\Psi}_{\mathrm{red}}$ is the reduced adjoint modal matrix of the system and it is noted that the two sets of modes are bi-orthogonal. At this point, it is important to note that as the two methods (ERA and ABPOD) are equivalent, that the modes can be calculated using the SVD outputs from the ERA method previously detailed; thus

leaving $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$ as the only additional data needed to calculate the reduced primal and adjoint modes respectively.

In ABPOD these modes can then be used along with the full order system matrices to calculate their reduced forms such that:

$$\begin{aligned}\widetilde{\mathbf{A}}_{red} &= \mathbf{\Psi}_{\mathrm{red}}{}^{*}\mathbf{A}\mathbf{\Phi}_{\mathrm{red}} \\ \widetilde{\mathbf{B}}_{red} &= \mathbf{\Psi}_{\mathrm{red}}{}^{*}\mathbf{B} \\ \widetilde{\mathbf{C}}_{red} &= \mathbf{C}\mathbf{\Phi}_{\mathrm{red}}\end{aligned} \tag{3.56}$$

Note that, for the purposes of the pressure reconstruction, only parts of the reduced primal modes ($\mathbf{\Phi}_{\mathrm{red}}$) are required to be calculated. Therefore, this is the only aspect of ABPOD that is required to be taken forward.

Taking Eqn. (3.55) and separating $\mathbf{\Phi}_{\mathrm{red}}$ into surface values (denoted by subscript sv) and all other values (denoted by subscript aov), and likewise sorting $\widetilde{\mathbf{X}}$ so that all the surface pressures are at the top, then it can written as:

$$\begin{bmatrix}\mathbf{\Phi}_{\mathrm{red}_{sv}} \\ \mathbf{\Phi}_{\mathrm{red}_{aov}}\end{bmatrix} = \begin{bmatrix}\widetilde{\mathbf{X}}_{sv} \\ \widetilde{\mathbf{X}}_{aov}\end{bmatrix} \mathbf{\Theta}_{red}\mathbf{\Sigma}_{red}{}^{-\frac{1}{2}} \tag{3.57}$$

therefore:

$$\mathbf{\Phi}_{\mathrm{red}_{sv}} = \widetilde{\mathbf{X}}_{sv}\mathbf{\Theta}_{red}\mathbf{\Sigma}_{red}{}^{-\frac{1}{2}} \tag{3.58}$$

We therefore see that the section of normal modes associated with surface pressures can be calculated using the time history of the surface pressure. Therefore massively reducing the amount of data needing to be stored in $\widetilde{\mathbf{X}}$. Next, considering an expansion in terms of a finite set of normal modes for an N dimensional system reduced to an r dimensional system:

$$\underline{\boldsymbol{x}}^{k} = \mathbf{\Phi}_{\mathrm{red}}\underline{\boldsymbol{x}}^{k}_{red} \tag{3.59}$$

where $\underline{\boldsymbol{x}}^{k}_{red}$is the reduced model approximation to the entire flow solution at time step k.

As with Eqn. (3.57), the same separation of surface values from all other values can be applied to Eqn. (3.59); thus it can be written as:

$$\begin{bmatrix}\underline{\boldsymbol{x}}^{k}_{sv} \\ \underline{\boldsymbol{x}}^{k}_{aov}\end{bmatrix} = \begin{bmatrix}\mathbf{\Phi}_{\mathrm{red}_{sv}} \\ \mathbf{\Phi}_{\mathrm{red}_{aov}}\end{bmatrix} \underline{\boldsymbol{x}}^{k}_{red} \tag{3.60}$$

therefore:

$$\underline{\boldsymbol{x}}_{sv}^{k} = \boldsymbol{\Phi}_{\mathrm{red}_{sv}} \underline{\boldsymbol{x}}_{red}^{k} \tag{3.61}$$

As $\underline{\boldsymbol{x}}_{red}^{k}$ is already calculated within the original ROM method, it can simply be stored for potential later use (at little computational cost). Then, should the pressure reconstruction be required, the impulse response solution at each time step is read in. As the surface pressure is the only part of the solution of interest for loads calculations, Eqn. (3.58) can then be used to obtain the reduced primal modes, and finally, Eqn. (3.61) can be used to calculate the output solution (for the surface values). It is worth noting that whilst the above refers to the surface pressures, it could equally be applied to such variables as the coefficient of pressure or skin friction.

The process as laid out above is for the original, step-up, style of ROM. However, it can be applied to step-down ROMs by substituting the impulse response solution with the steady solution minus those of the sharp-edged gust. The output solutions are then corrected by simply adding the steady values.

Finally it is important to note that while the method is valid for MIMO systems it has only been applied to single-input, single-output (SISO) systems because of the difficulties of stabilising MIMO systems for small numbers of time steps. Therefore, the output surface pressures will vary with both the original choice in output from the underlying ROM, as well as reduced ROM size. This is because the reconstruction process will not necessarily produce an identical set of pressures to those obtained via full order CFD. Instead, the reconstruction process will produce a set of surface pressures based on the modes included. For small ROM sizes, there will only be a small number of modes, which will be closely tied to the force coefficient output of the underlying ROM. In these cases, the reconstructed surface pressures would be expected to integrate to match said force coefficient output; but they may not fully match the full order CFD pressures and thus may not integrate to give the expected value for other force coefficients. In a larger ROM, more modes are included and so the closer reconstructed surface pressures would match those from the full order CFD.

4. ROM Applied to Inviscid Wing Model

To test the ROM methods described in this thesis, the rigid Future Fast Aeroelastic Simulation Technologies (FFAST) wing was used. As the author was not involved in the development of the model, therefore it would be distracting from the work carried out to go into too much detail; however, for completeness, a brief overview is provided.

The model consists of an aircraft wing, split in half, and modelled within CFD with a symmetry plane on the wing root accurately; as shown in Figure 4.1. The flow simulations performed for this case were all inviscid.

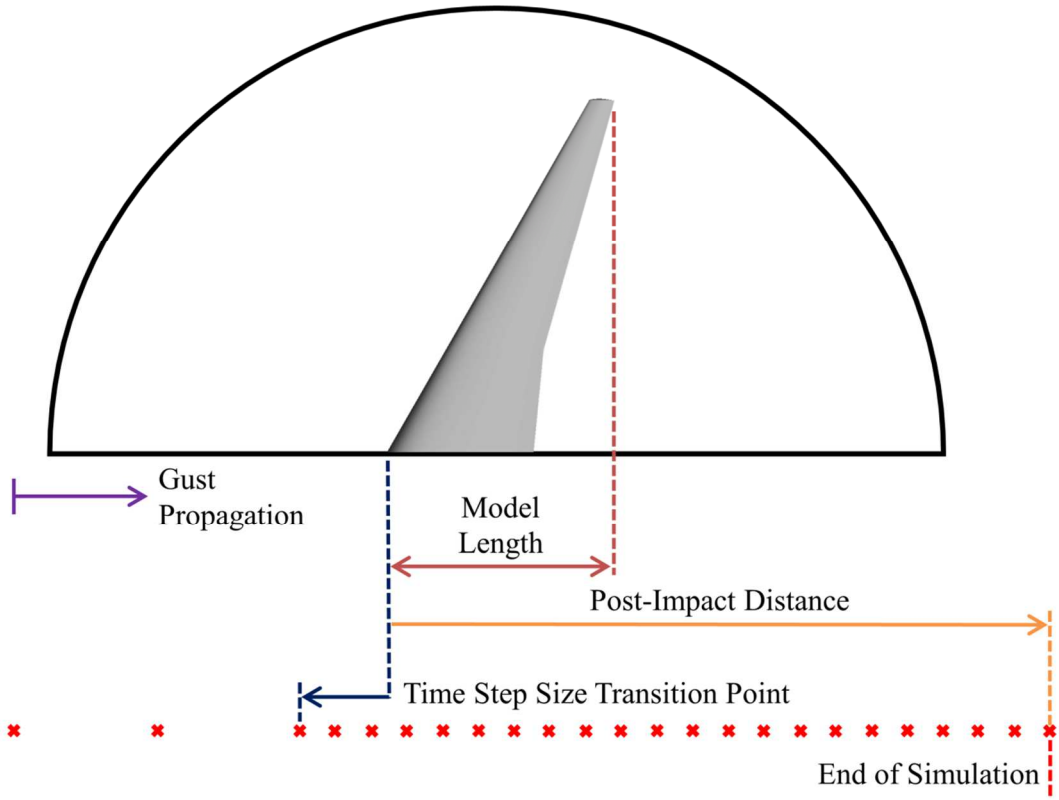

Figure 4.1. Representation (not to scale) of the top down view of the domain for the FFAST wing geometry.

The aerodynamic model consisted of an unstructured mesh throughout a hemisphere domain (see Figures 4.2 and 4.3). The mesh was made up of 2,714,778 tetrahedron cells and had 95,250 triangular surface elements. As this was an inviscid simulation, the Euler equations acted as the governing equations being solved for (see Section 2.3).

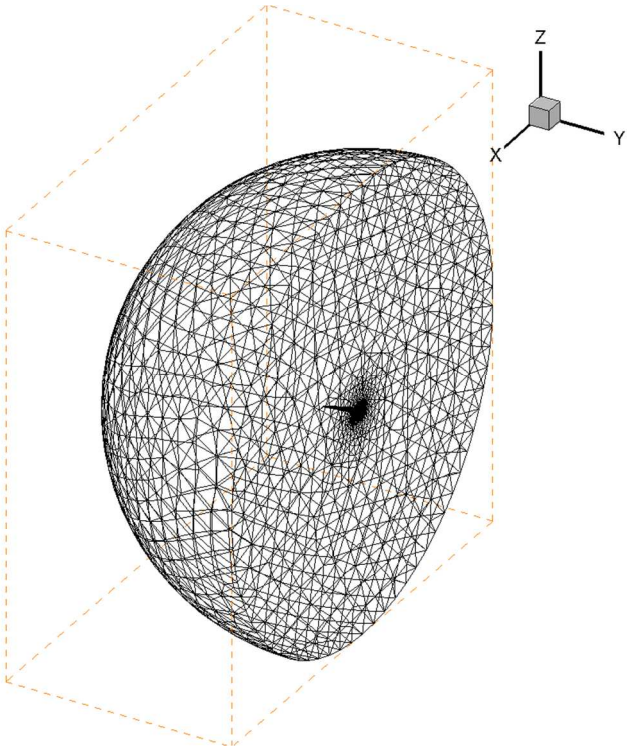

Figure 4.2. Domain mesh for the wing model.

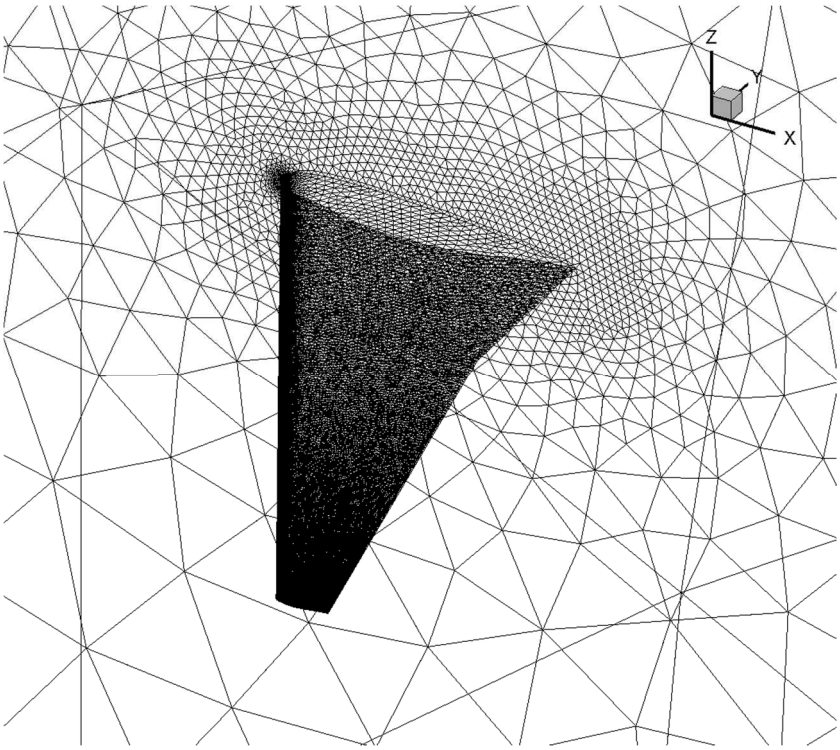

Figure 4.3. Surface mesh for the wing model.

All gusts modelled were '1-cosine' vertical gusts, as defined by the CS-25 regulations [2] as detailed in Section 1.2.2 For the FFAST wing, no gust alleviation was applied, and all flight points are assumed to be non-VD (design diving speed) cases. Four gust lengths at each of five flight points were explored for the FFAST wing (see Table 3).

Table 3. Details of gusts explored for the FFAST wing.

Flight Point	Mach Number	Altitude (ft)	Altitude (m)	Gust 1 18m Velocity (ms^{-1})	Gust 2 80m Velocity (ms^{-1})	Gust 3 146m Velocity (ms^{-1})	Gust 4 214m Velocity (ms^{-1})
1	0.500	0	0	11.299	14.488	16.016	17.070
2	0.735	0	0	11.299	14.488	16.016	17.070
3	0.800	0	0	11.299	14.488	16.016	17.070
4	0.800	29,995	9,142.476	7.321	9.388	10.378	11.061
5	0.800	43,000	13,106.400	5.973	7.659	8.466	9.023

4.1. Baseline ROM Method

The discrete method laid out in Chapter 3 shall be referred to as the baseline ROM method and will used as a starting point for further development. Thus, to create the ROM, a series of steady simulations are carried out for a range of angles of attack (which will be referred to as the polar sweep). Additionally, two sharp-edged gusts, one with a magnitude of $1ms^{-1}$ and one with a magnitude of $2ms^{-1}$, are simulated using small, constant time steps; these simulations are also run until the gust is very far downstream from the test geometry. From the sharp-edged gusts, the pulse response of the system is identified. This pulse response is then used to create the Hankel matrix via ERA. Next, Singular Value Decomposition is used to perform the system reduction and obtain the $\mathbf{A}_{red}$, $\mathbf{B}_{red}$ and $\mathbf{C}_{red}$ matrices; which are only valid at the flight point (Mach number and altitude) that the setup simulations were run at. As such, a new ROM must be created for each flight point; with each given ROM capable of being used to obtain the system response to any gust at that flight point.

To improve the accuracy of the system when non-linearities increase, a method known as steady state correction is applied; as laid out in Section 3.3. To calculate the scaling matrix (**S**), first the linear gradients must be calculated for each force coefficient. This is done via the simple mathematical equation:

$$\frac{dC_F}{dv_g} \sim \frac{C_{F_2} - C_{F_1}}{v_{g_2} - v_{g_1}} \tag{4.1}$$

where C_F is any force coefficient (e.g. the coefficient of lift), v_g is the gust velocity and subscript 1 and 2 denote two arbitrary gust magnitudes.

4.1.1. Results

In order to explore the accuracy of the baseline ROM method, and to allow for quick development, one flight point was initially studied. After a preliminary study, a ROM size of 20 was found to give a good balance between minimising computational cost and keeping a high level of accuracy. As such, the baseline ROM method with a ROM size of 20, was applied to the four gust cases of flight point 2.

It is worth noting that this ROM, as with all the ROMs presented within this thesis (unless stated otherwise), are SISO and thus $\boldsymbol{y}$ is either $\hat{C}_l$ *or* $\hat{C}_m$.

Figures 4.4-4.11 show very close matches to the full order CFD simulations (particularly for the coefficient of lift) with only the longer gust lengths exhibiting any noticeable disagreement between the two. As this level of accuracy is so high, it was decided to focus development on improving the computational costs associated with building the ROM, whilst maintaining the high level of accuracy, rather than on attempting to reduce the error margin further; although this continued to be a secondary objective.

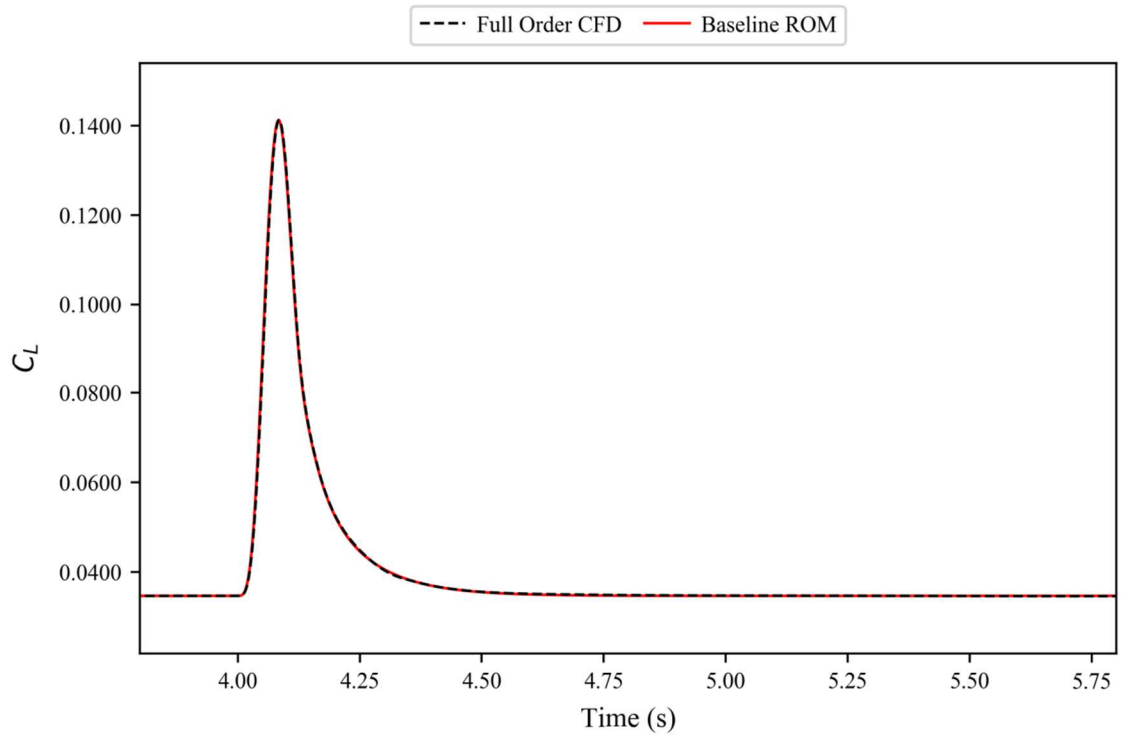

Figure 4.4 Baseline ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

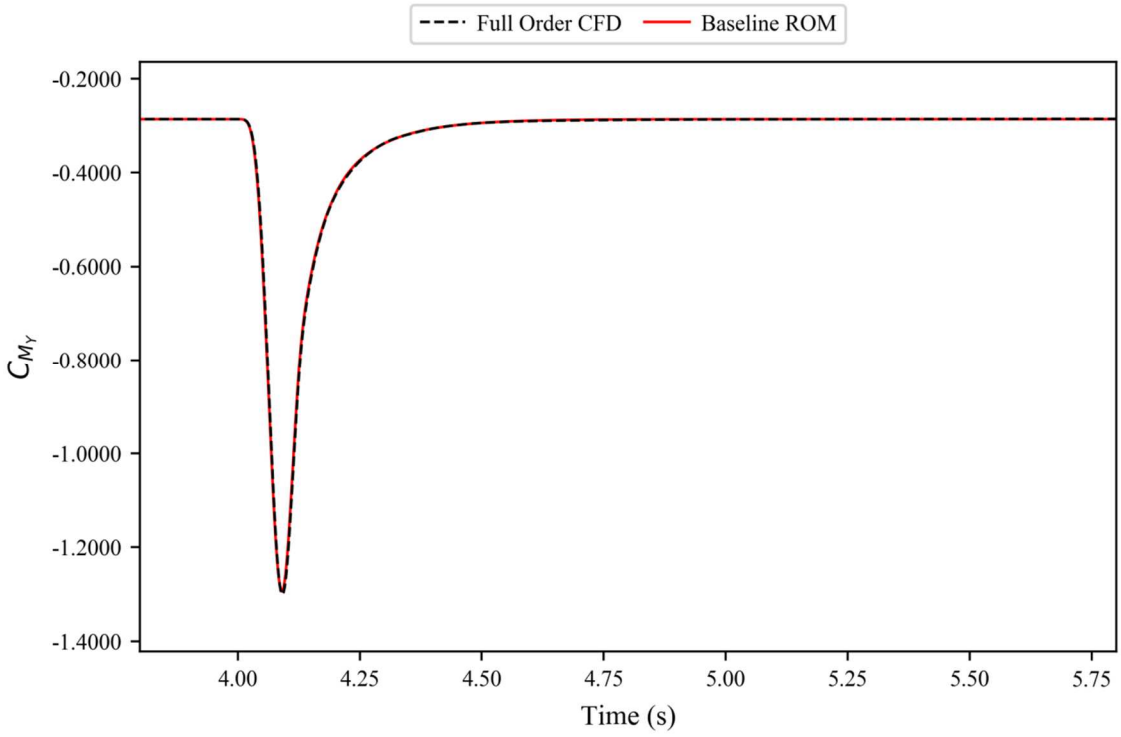

Figure 4.5 Baseline ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

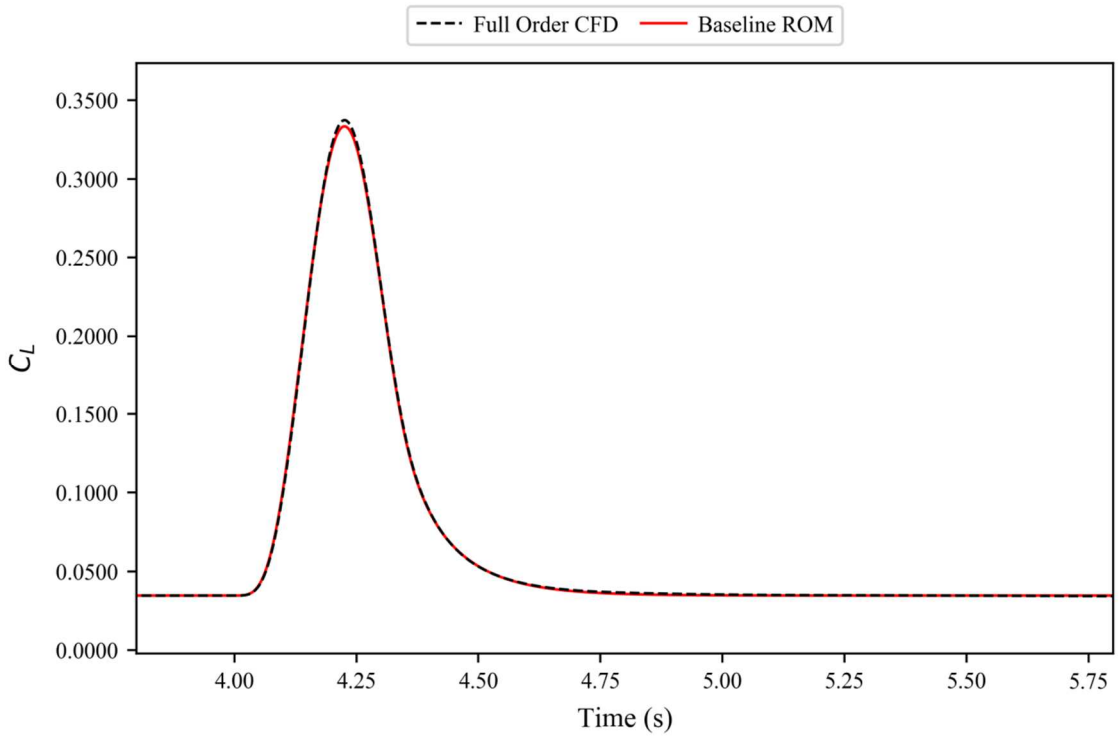

Figure 4.6 Baseline ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

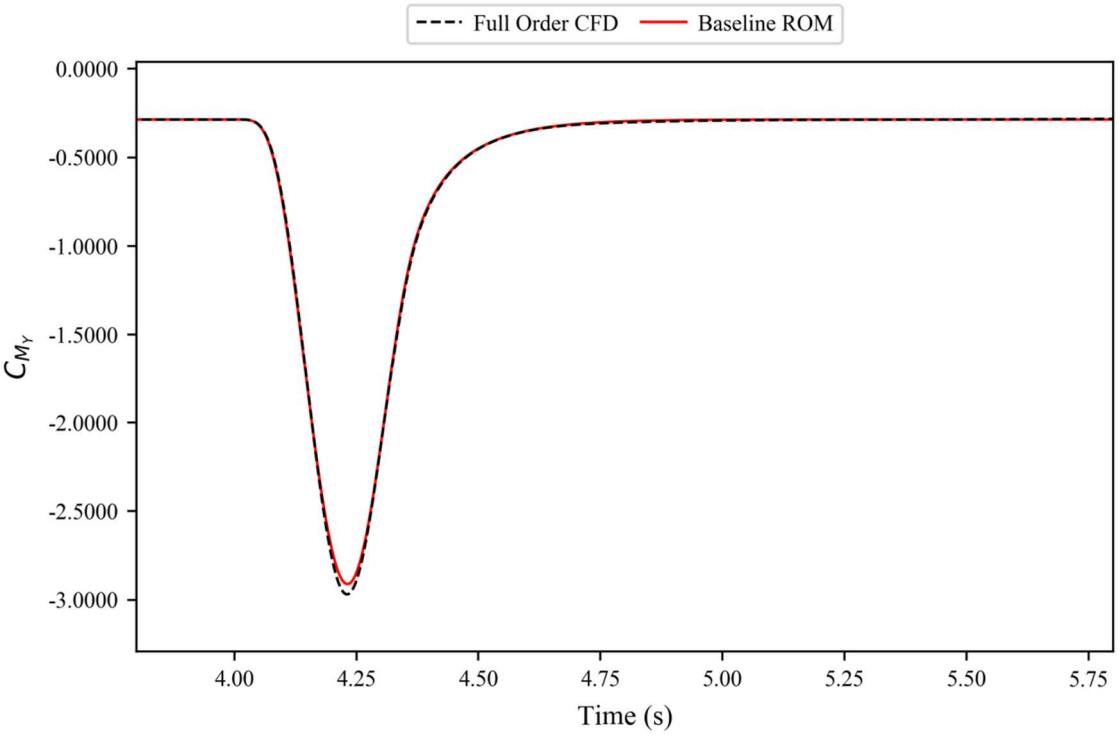

Figure 4.7 Baseline ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

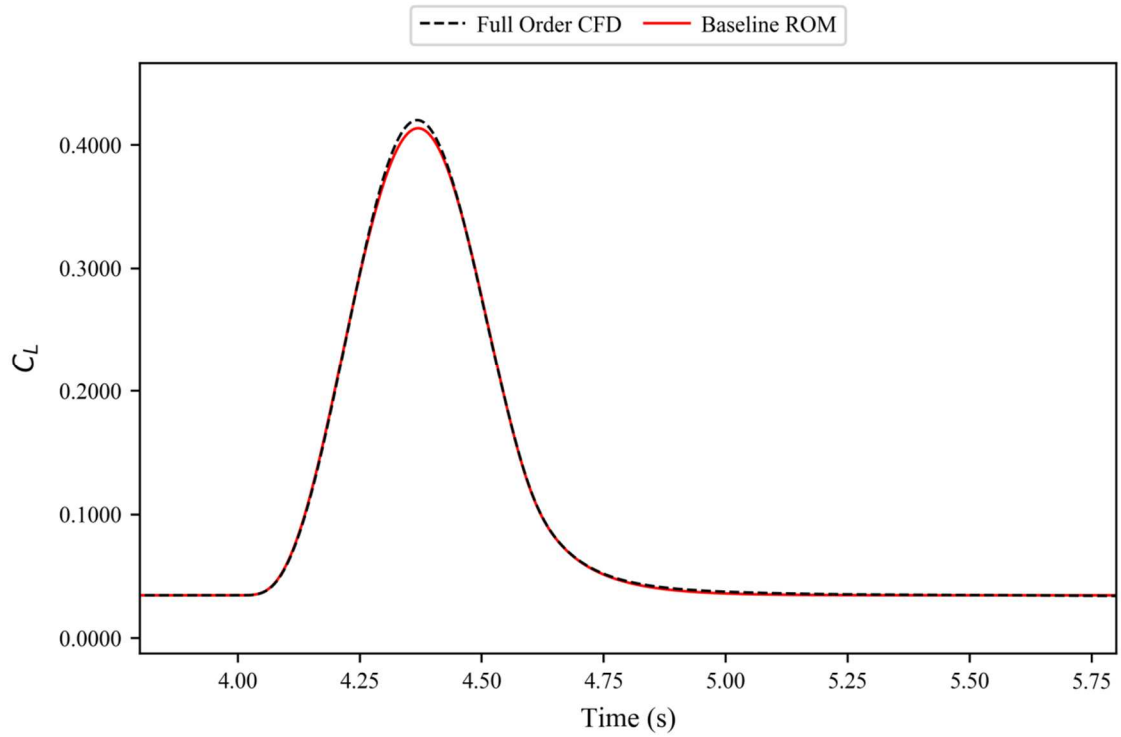

Figure 4.8 Baseline ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

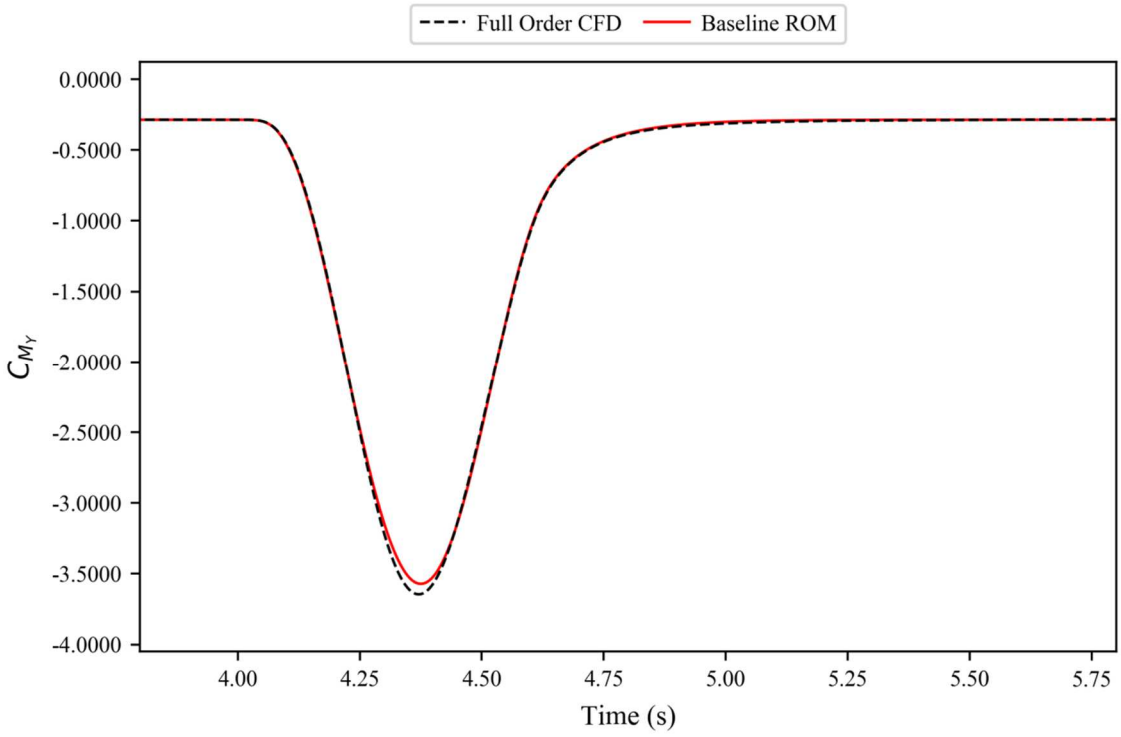

Figure 4.9 Baseline ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

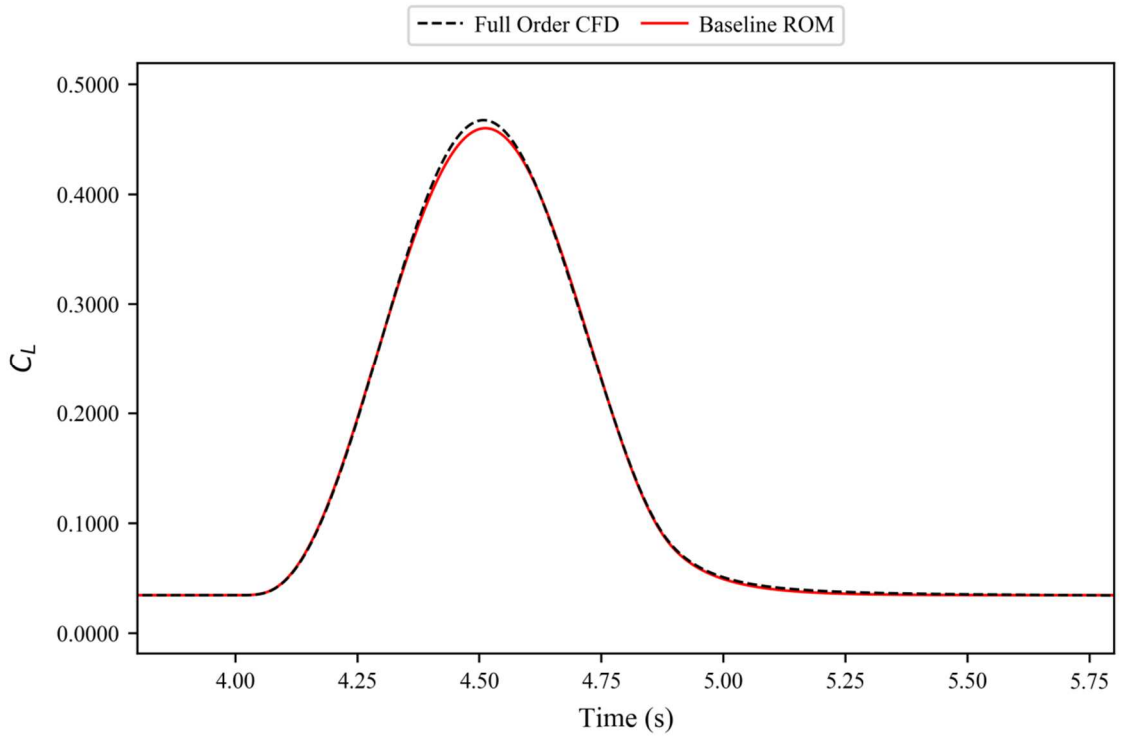

Figure 4.10 Baseline ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

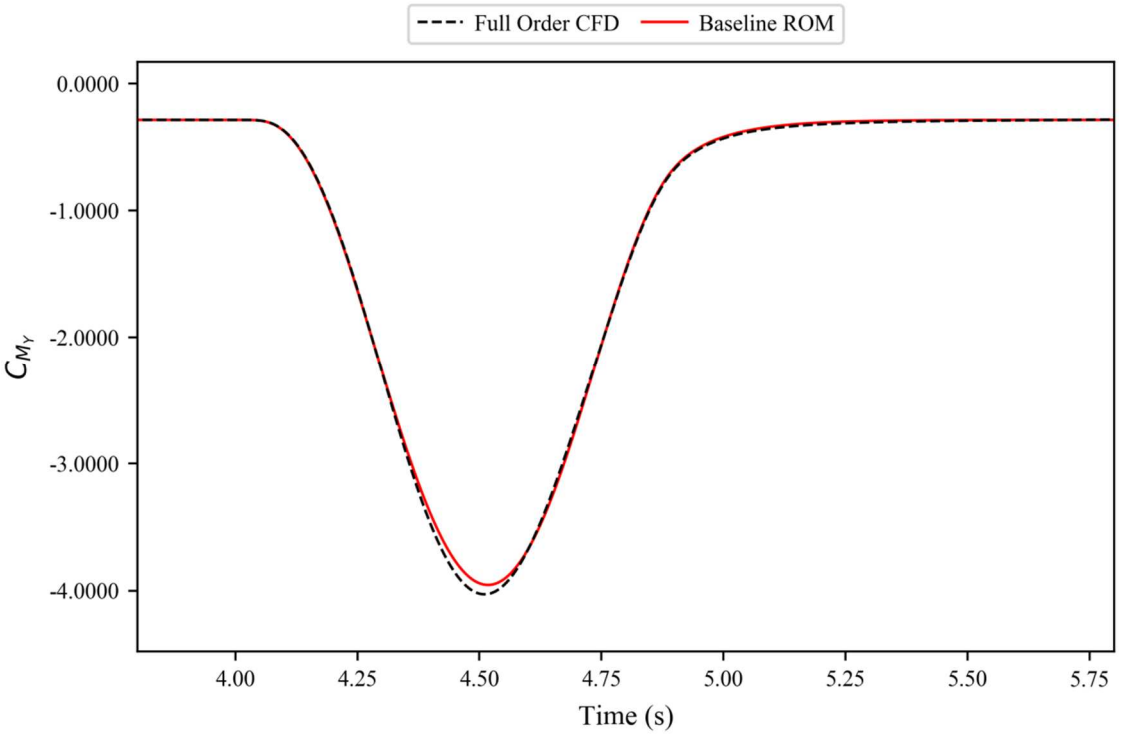

Figure 4.11 Baseline ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

It should be noted that even this baseline ROM method offers significant computational savings compared with the full order simulations. To produce a ROM model with this method, the computational resources are approximately equal to running one full order

CFD simulation. However, the true savings from the ROM come when modelling multiple gust cases for a given flight point. Once built, the computational cost of calculating the system response to any given gust is near negligible (capable of being calculated in seconds on a typical desktop computer). As an example, if building the ROM is comparable in computational cost to running a single '1-cosine' CFD simulation, then for the ROM the total computational cost (including building the ROM) of obtaining the system response of four gusts is approximately only a quarter of that to obtain the same via full order CFD. This offers potentially very large savings when compared to full order CFD simulations when numerous gusts responses are desired.

4.2. Single Sharp-Edged Gust Method

With the primary objective of reducing the computational cost of building the ROM, the first area identified as potentially offering large room for improvement was the way in which the pulse response was calculated.

The baseline ROM method required two sharp-edged gusts to be run, one with twice the peak gust velocity of the other, to produce the pulse response. If the larger sharp-edged gust has a peak gust velocity that is suitably small, then the Volterra theory based method of building the linear pulse response (see Section 3.2.2) can be simplified.

For the baseline ROM method, the smaller of the sharp-edged gusts used had a peak velocity of $1ms^{-1}$. Taking this gust to be the linear response means only one gust simulation is needed.

4.2.1. Results

As with the baseline ROM method, we can assess the accuracy of the constructed ROM (with a ROM-size of 20) by comparing it to full order CFD results for the same four gusts at flight point 2.

It is immediately apparent from Figures 4.12-4.19 that the single sharp-edged gust based ROM method produces results that are extremely similar to those of the baseline ROM method. This shows that approximating the linear response as the smaller gust is

valid and produces results with no notable loss in accuracy (relative to the base ROM). However, as a result of only using one sharp edged gust, the computation cost of building the ROM is greatly reduced (this is equal to approximately a 40% reduction when compared to the base ROM; although this will vary depending on a variety of factors, such as CFD convergence criteria, etc.).

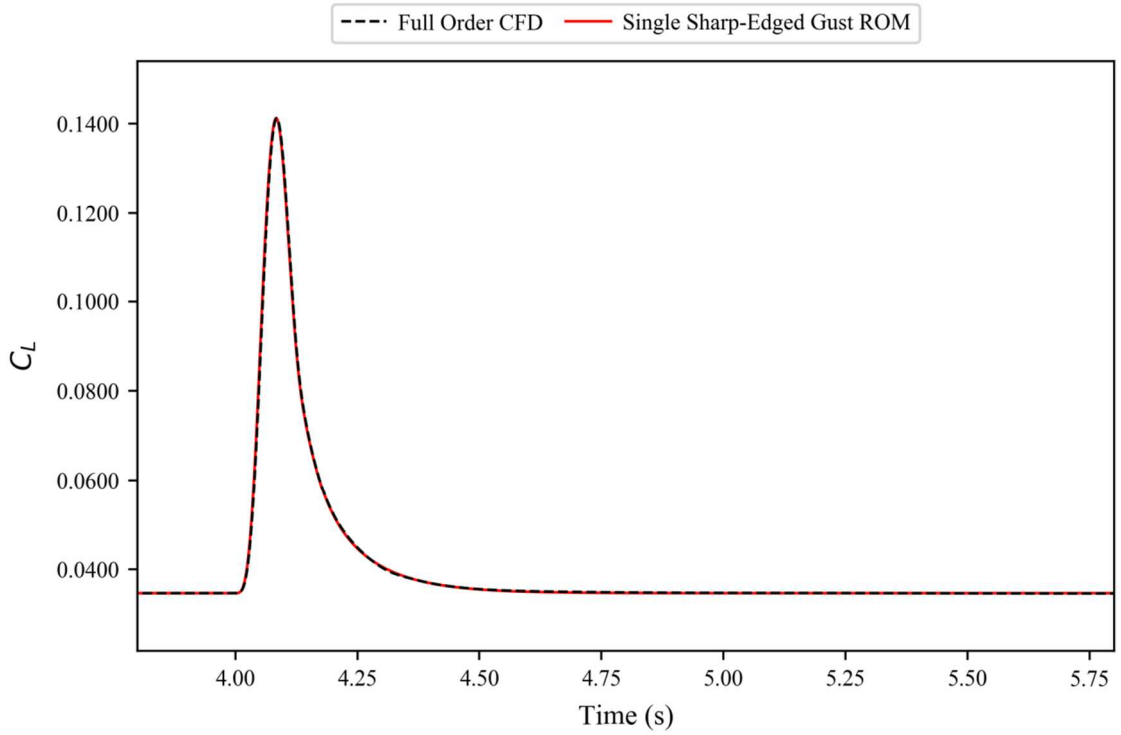

Figure 4.12 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

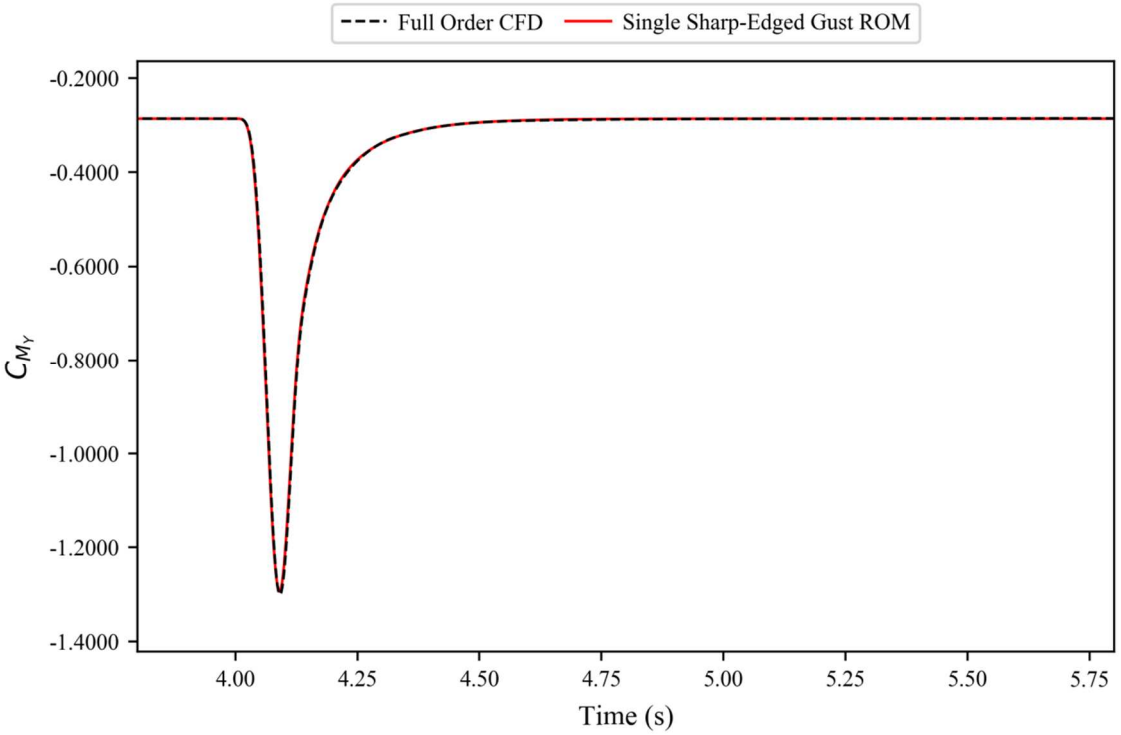

Figure 4.13 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

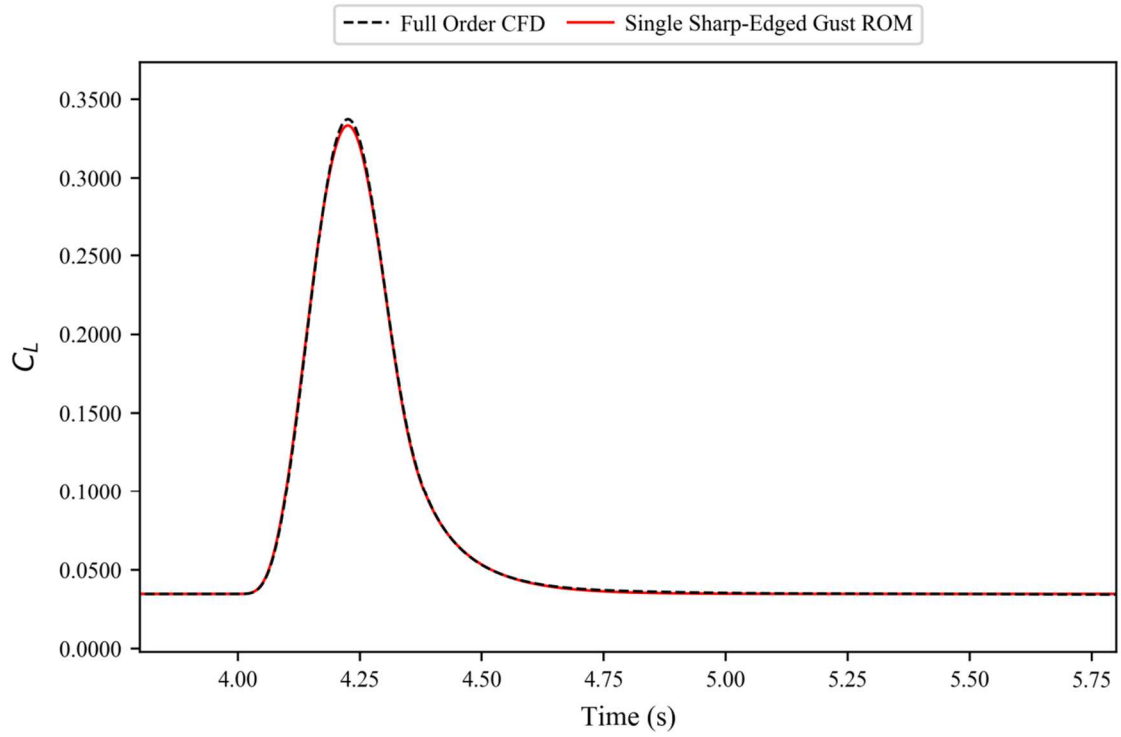

Figure 4.14 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

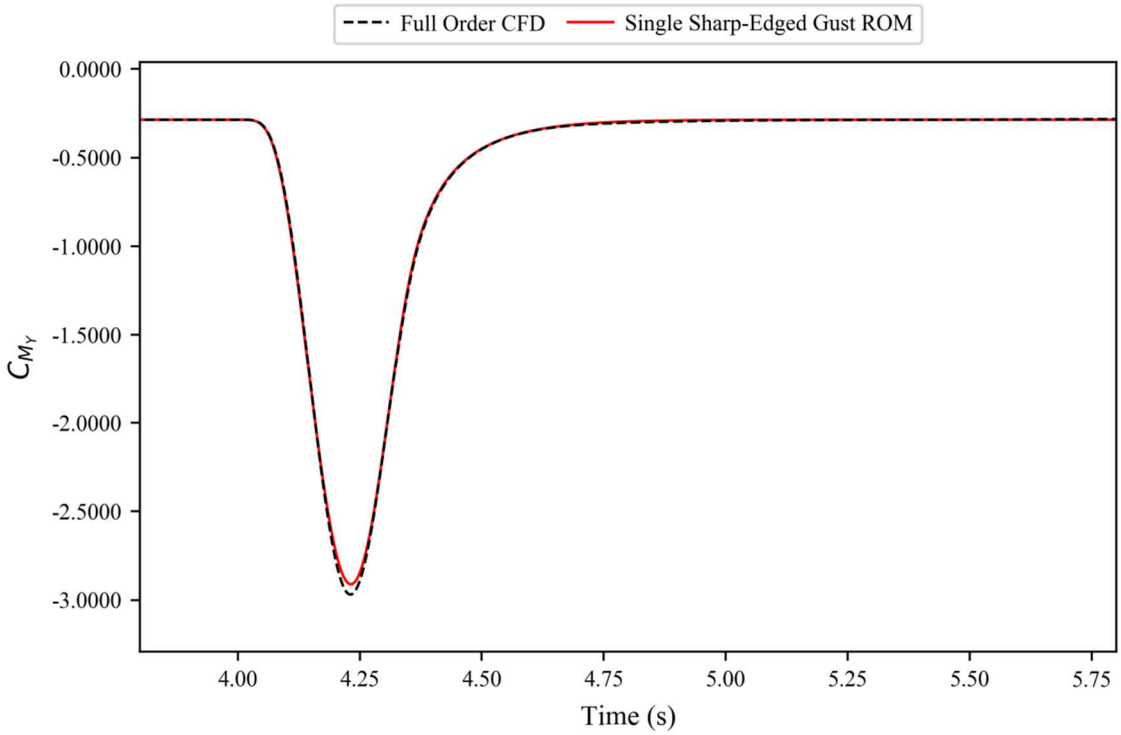

Figure 4.15 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

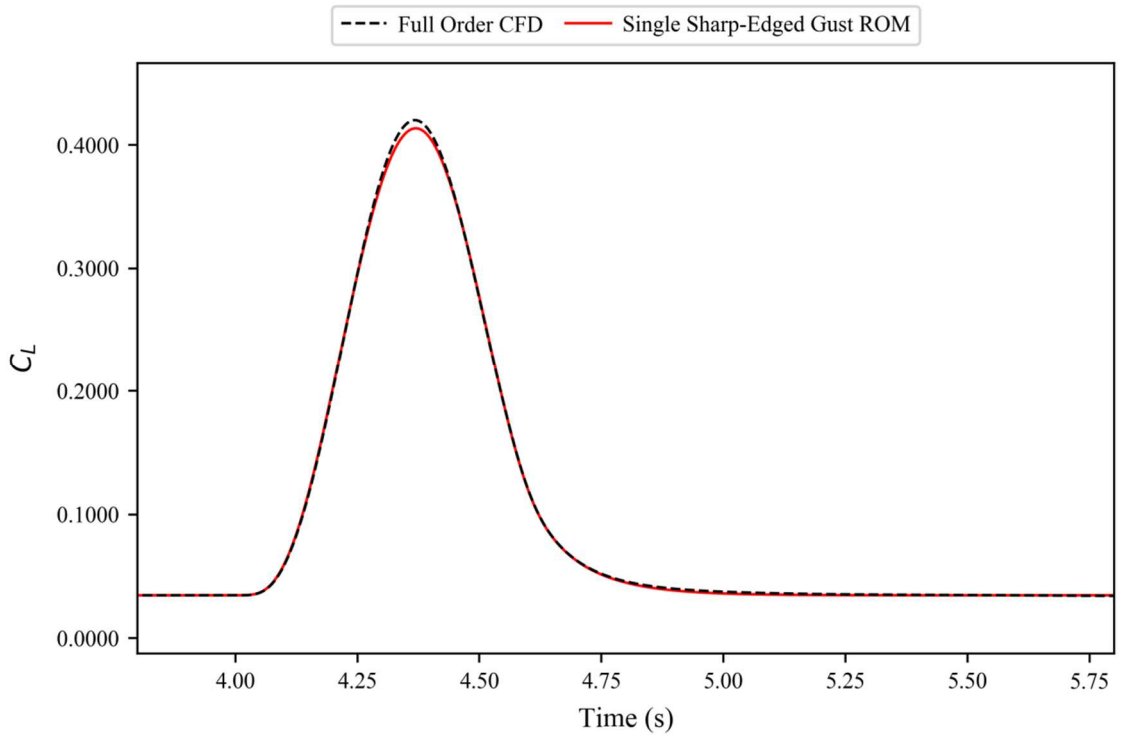

Figure 4.16 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

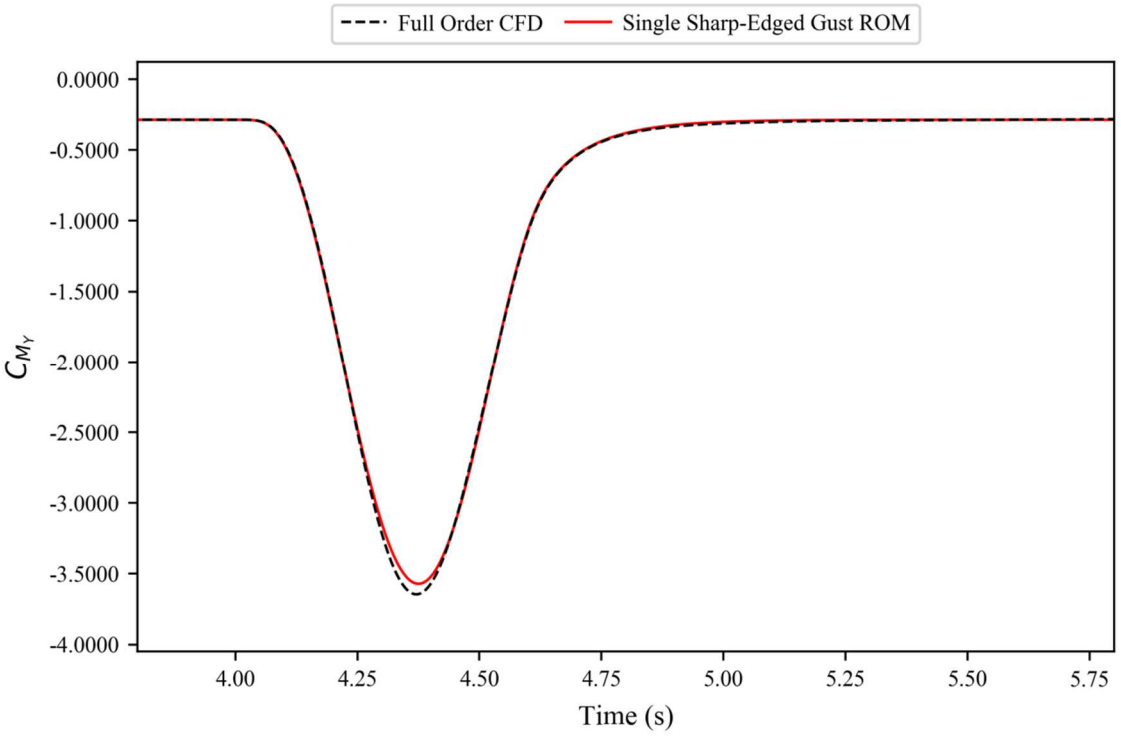

Figure 4.17 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

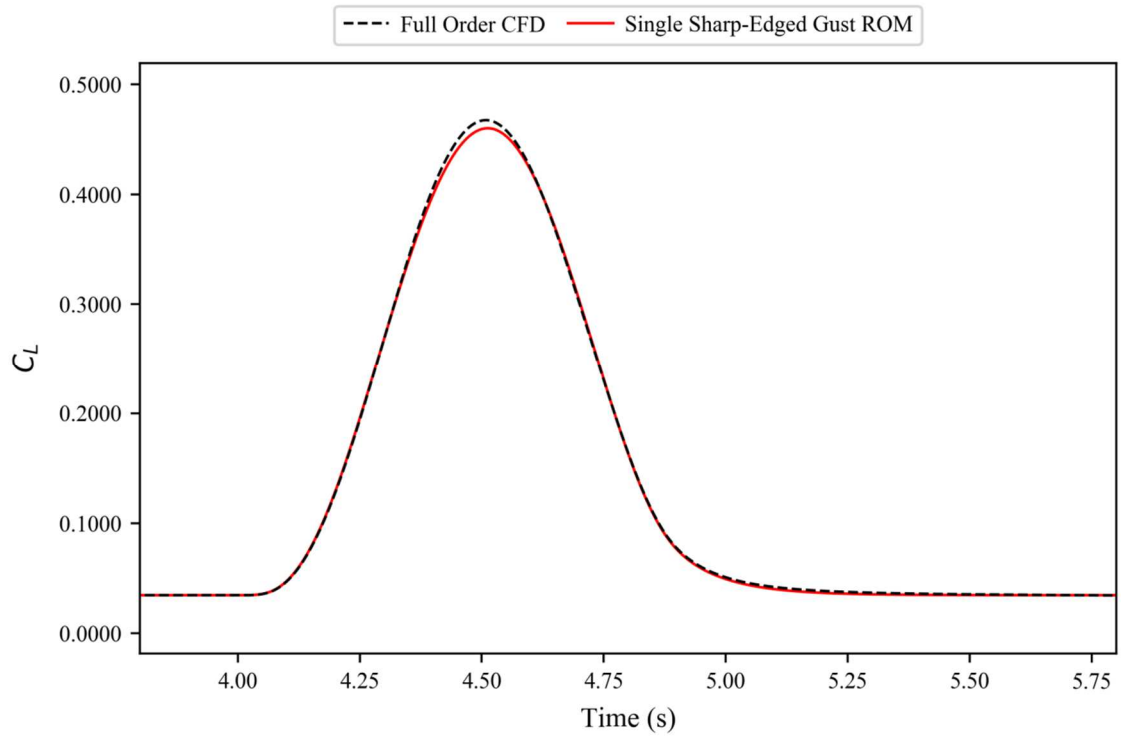

Figure 4.18 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

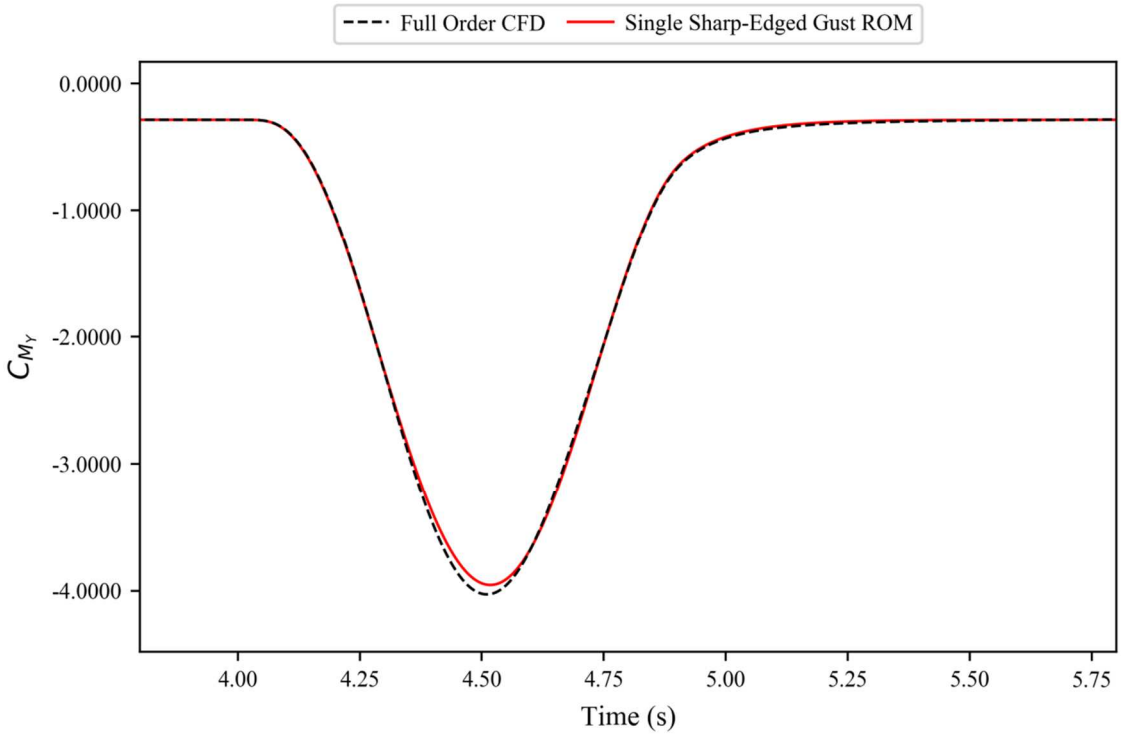

Figure 4.19 Single sharp-edged gust based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

4.3. Variable Time-Step Based ROM Method

Both the baseline ROM method and the single sharp-edged gust based ROM method were built using sharp-edged gusts that had a constant time step size. This time step was necessarily small in order to ensure the results of the CFD were suitably accurate. However, this is only required when the gust is interacting with the model. Therefore, having large time steps to bring the gust into the computational domain, before changing to smaller time steps for the remainder of the simulation, may offer significant savings without having a notable, detrimental impact on the accuracy of the ROM (see Figure 4.1).

Whilst most of system response would be captured when the gust first impacts the model, there may be some parts of the response that are generated shortly before this point. Therefore, it was necessary to explore the effect that the location of the transition point (from large to small time steps) may have. To perform this study, multiple sharp-edged gust simulations were set up. For each one the transition point was located in a different position; measured in terms of mean aerodynamic chord (MAC) lengths upstream from the farthest forward part of the geometry.

4.3.1. Varying Time Step Size Transition Point for the Baseline ROM

Figures 4.20 and 4.21 show that the location of the transition point does not greatly affect the system response. However, when the transition occurs on the leading edge itself, a slight degradation in accuracy can be noticed during the initial deviation from the system's steady state; this is more visible in the close ups of these plots, shown in Figures 4.22 and 4.23.

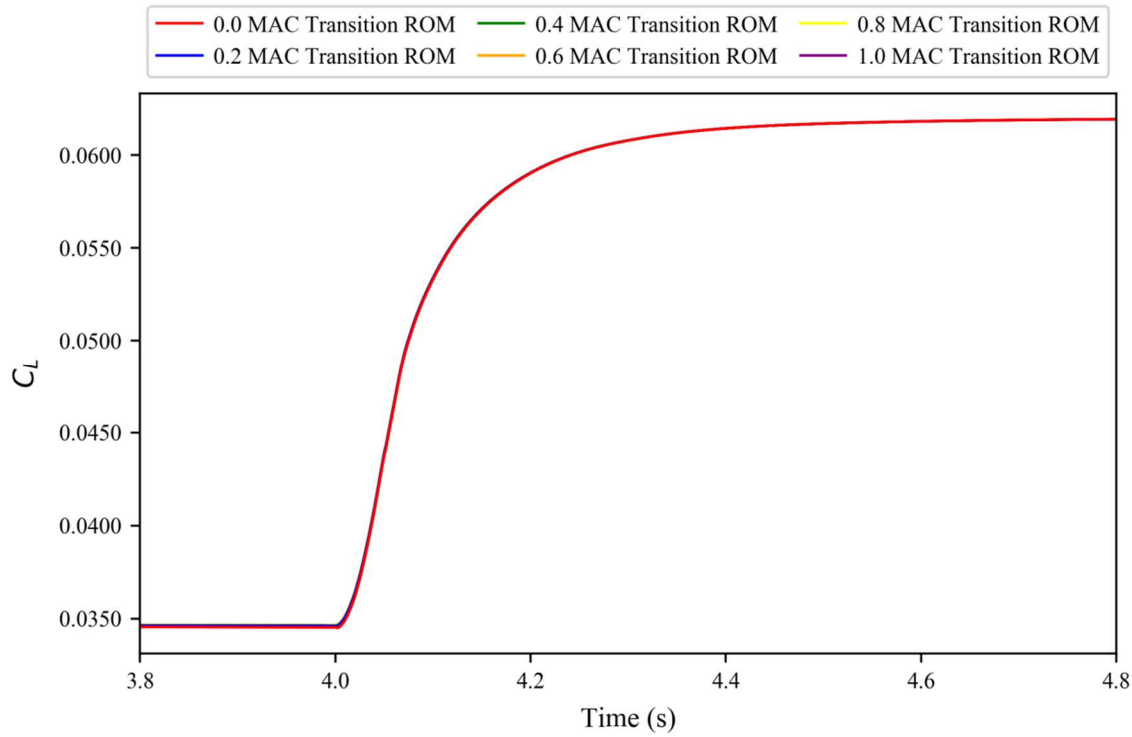

Figure 4.20 Coefficient of lift of the FFAST wing for 1ms^{-1} magnitude sharp-edged gusts transitioning from large to small time steps at different distances ahead of the leading edge.

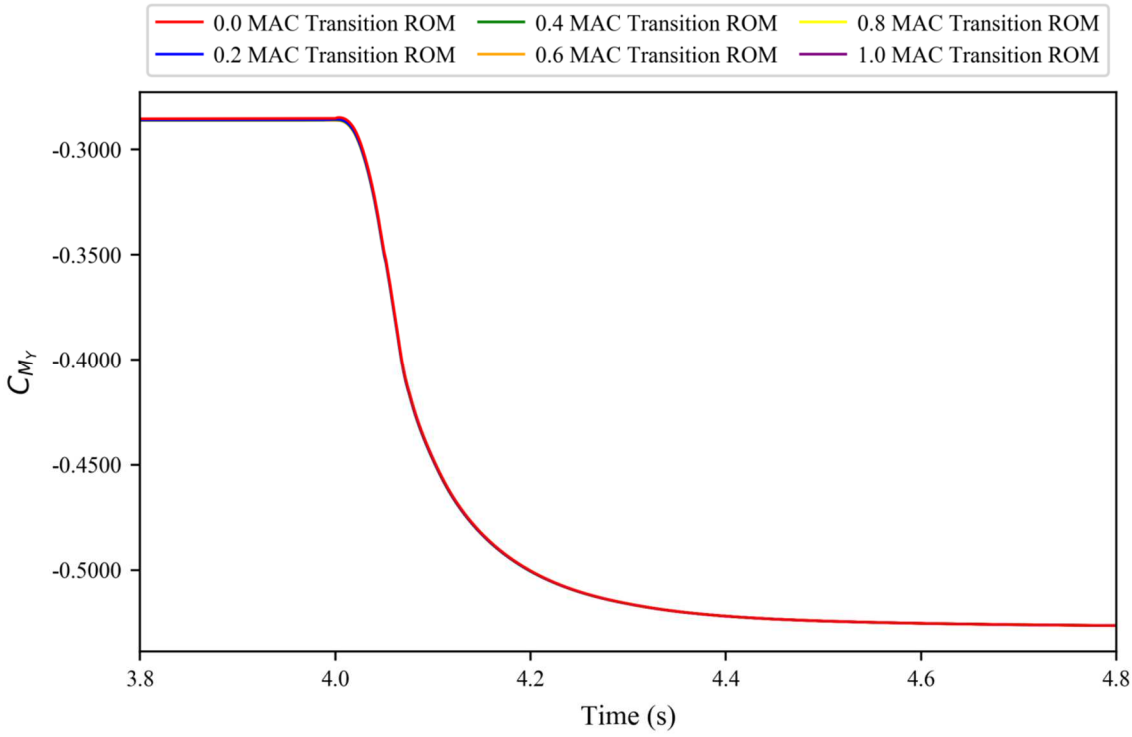

Figure 4.21 Coefficient of pitching moment of the FFAST wing for 1ms-1 magnitude sharp-edged gusts transitioning from large to small time steps at different distances ahead of the leading edge.

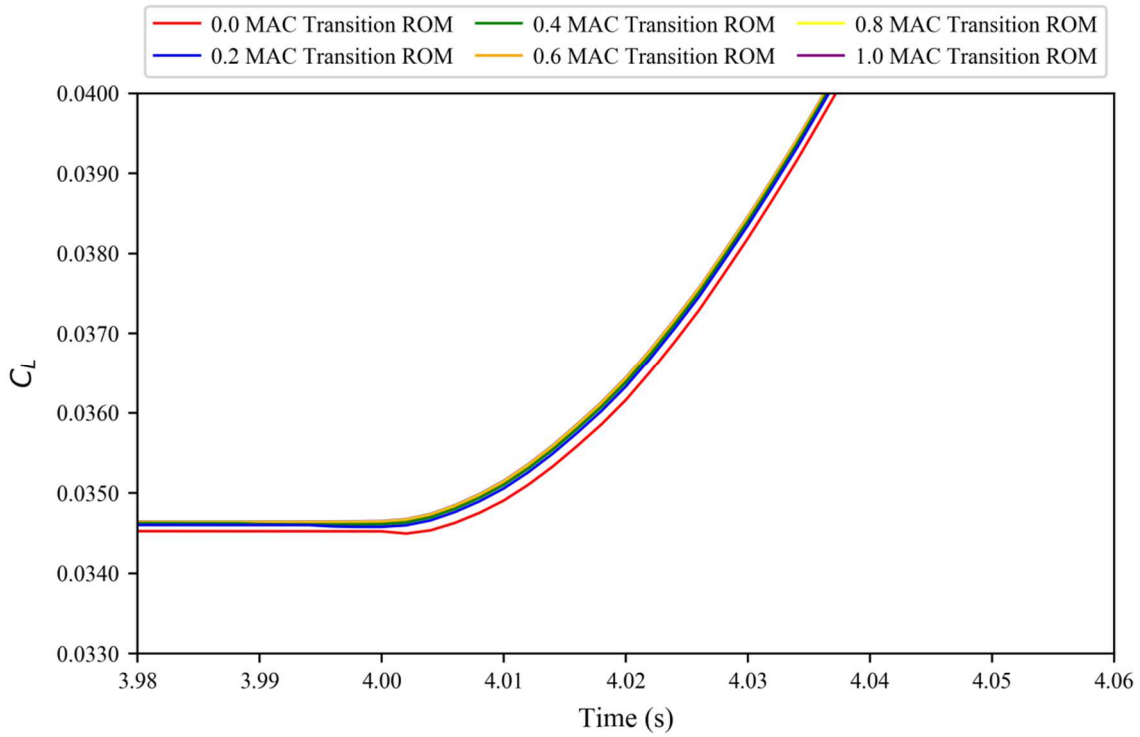

Figure 4.22 Initial change in the coefficient of lift of the FFAST wing for 1ms^{-1} magnitude sharp-edged gusts transitioning from large to small time steps at different distances ahead of the leading edge.

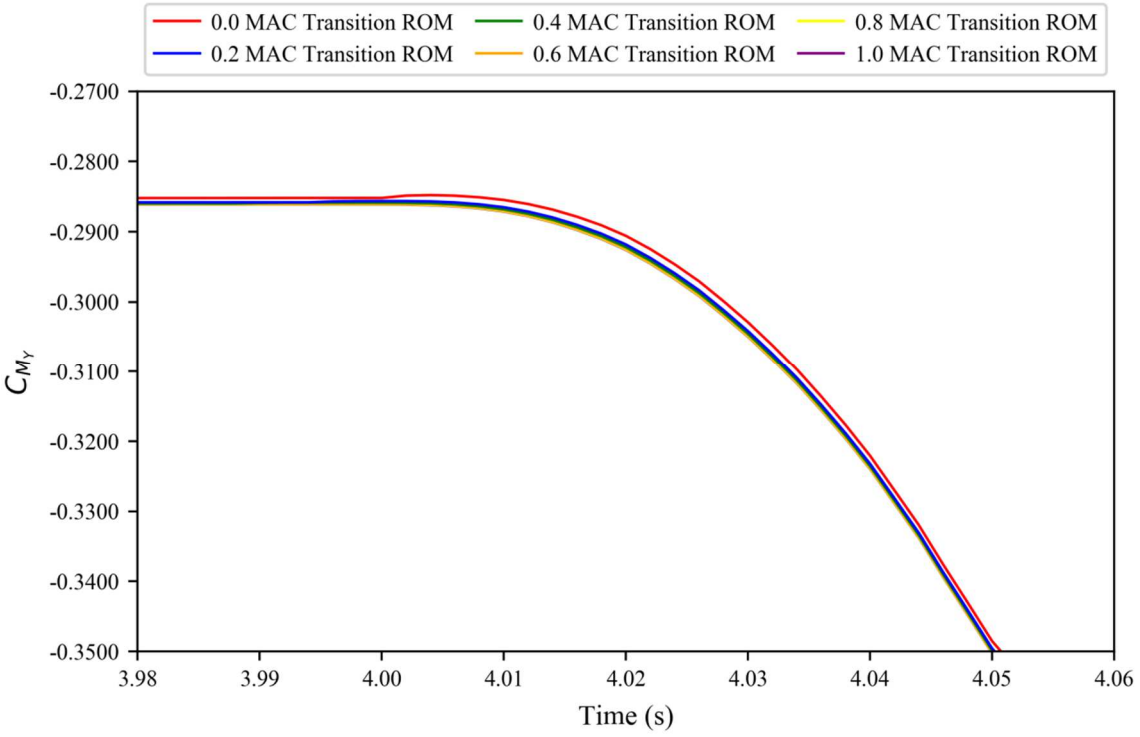

Figure 4.23 Initial change in the coefficient of pitching moment of the FFAST wing for 1ms-1 magnitude sharp-edged gusts transitioning from large to small time steps at different distances ahead of the leading edge.

4.3.2. Results

Again, the accuracy of the new ROM method (with varying time step size transition points) can be assessed by applying it to the gusts from flight point 2 for the FFAST wing; all with a ROM-size of 20.

Figures 4.24-4.31 show that, just as with the change to the single sharp-edge gust, the accuracy of the ROM has been maintained with respect to the baseline ROM method; with the results still being a very close match to those of the full order CFD.

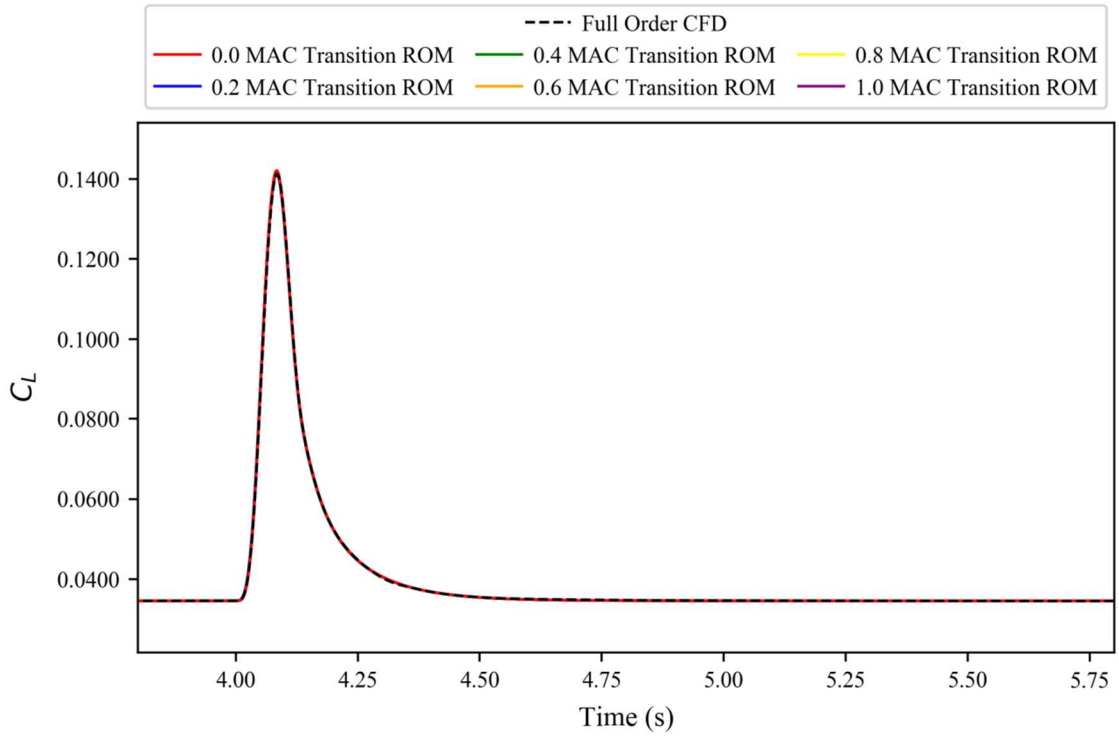

Figure 4.24 Variable time-step based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

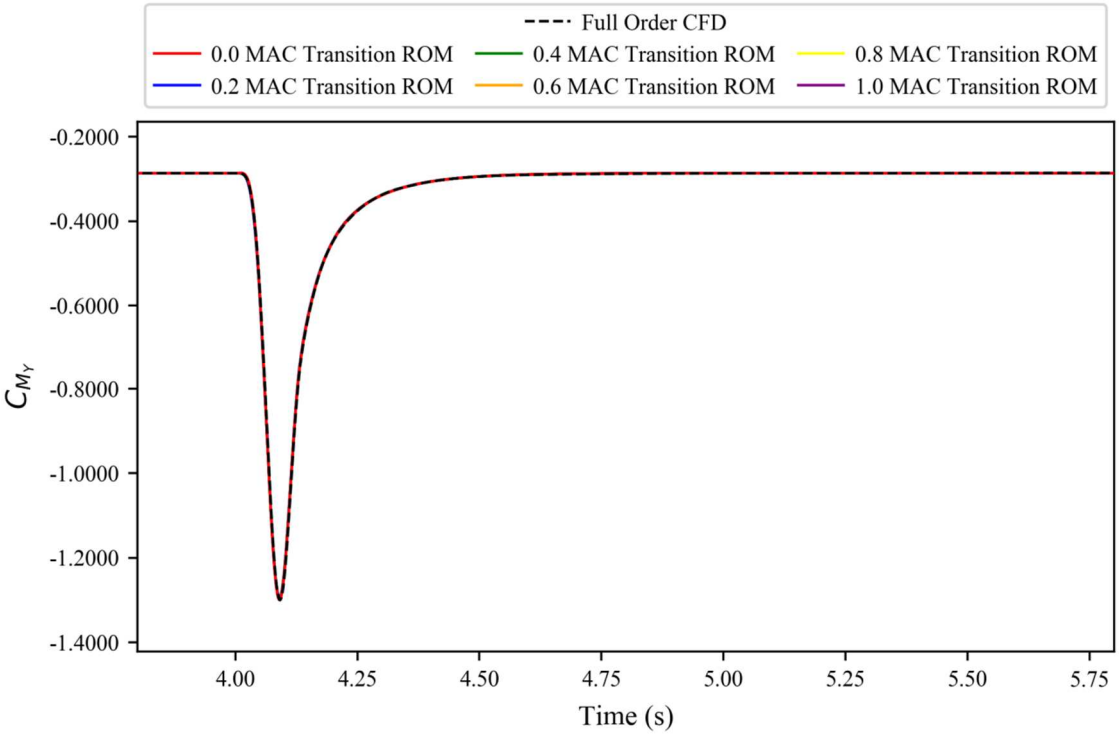

Figure 4.25 Variable time-step based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

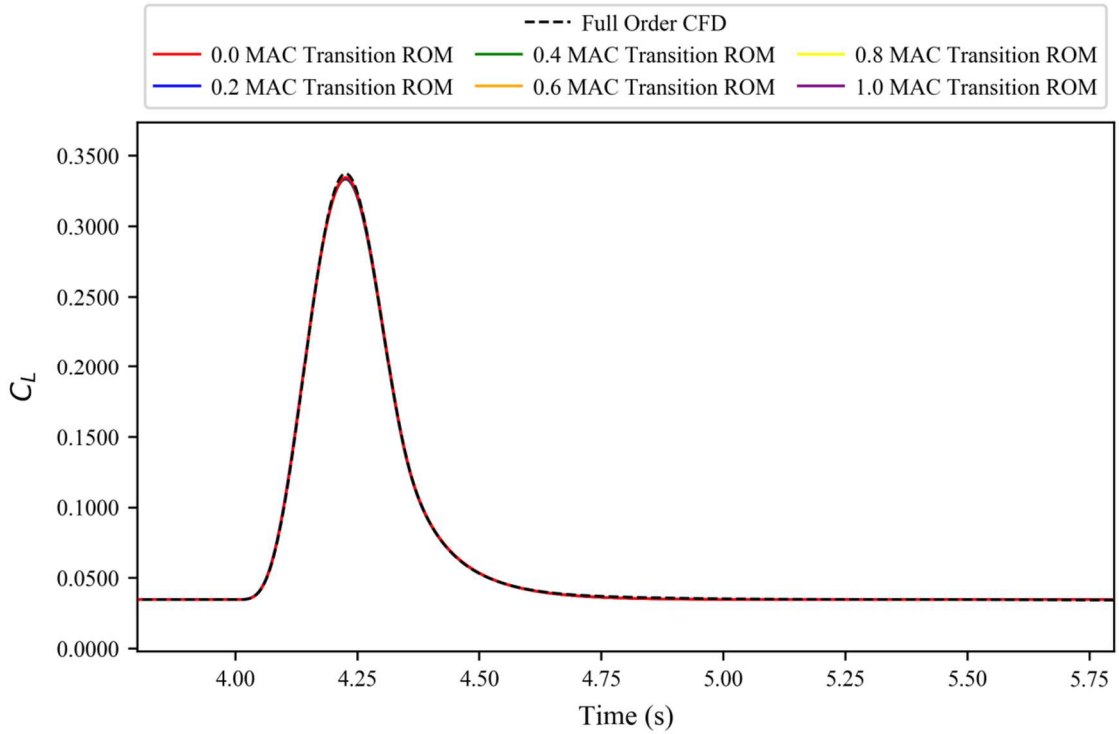

Figure 4.26 Variable time-step based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

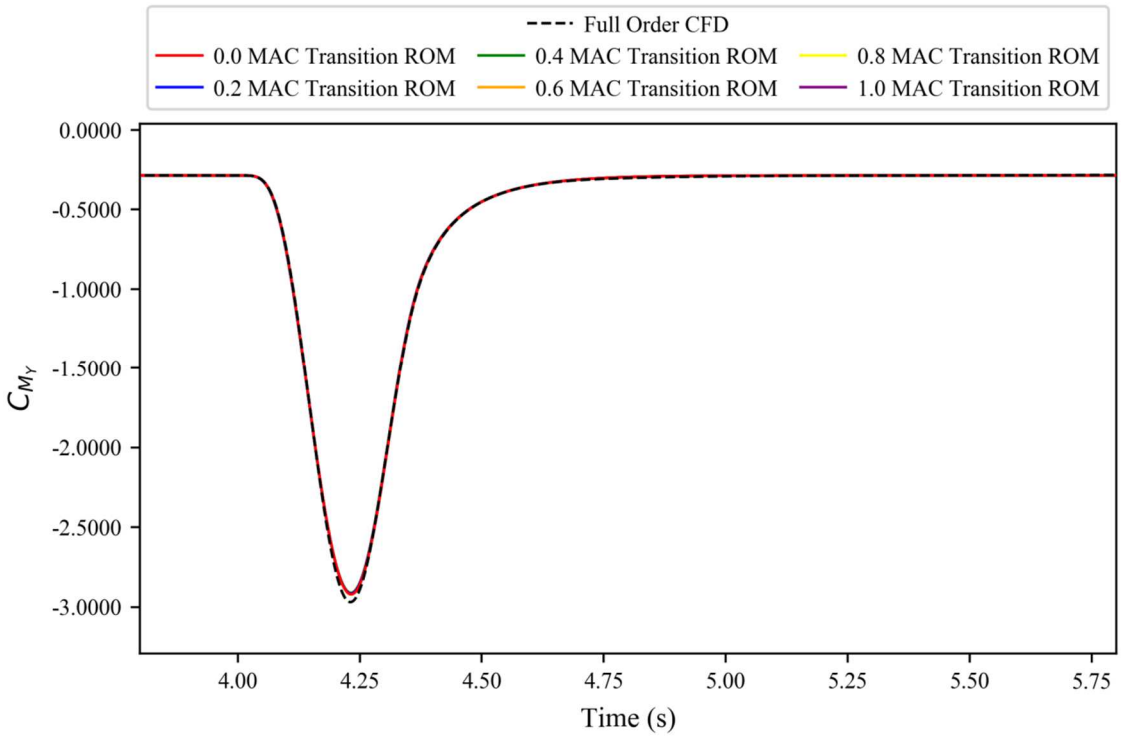

Figure 4.27 Variable time-step based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

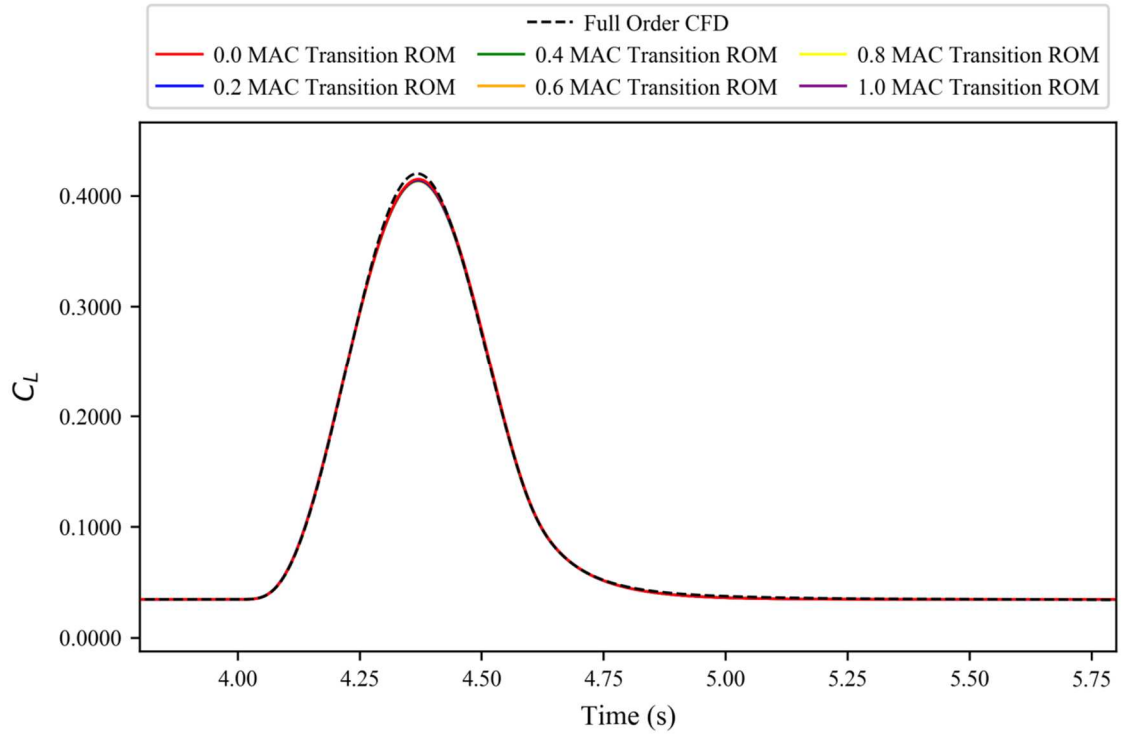

Figure 4.28 Variable time-step based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

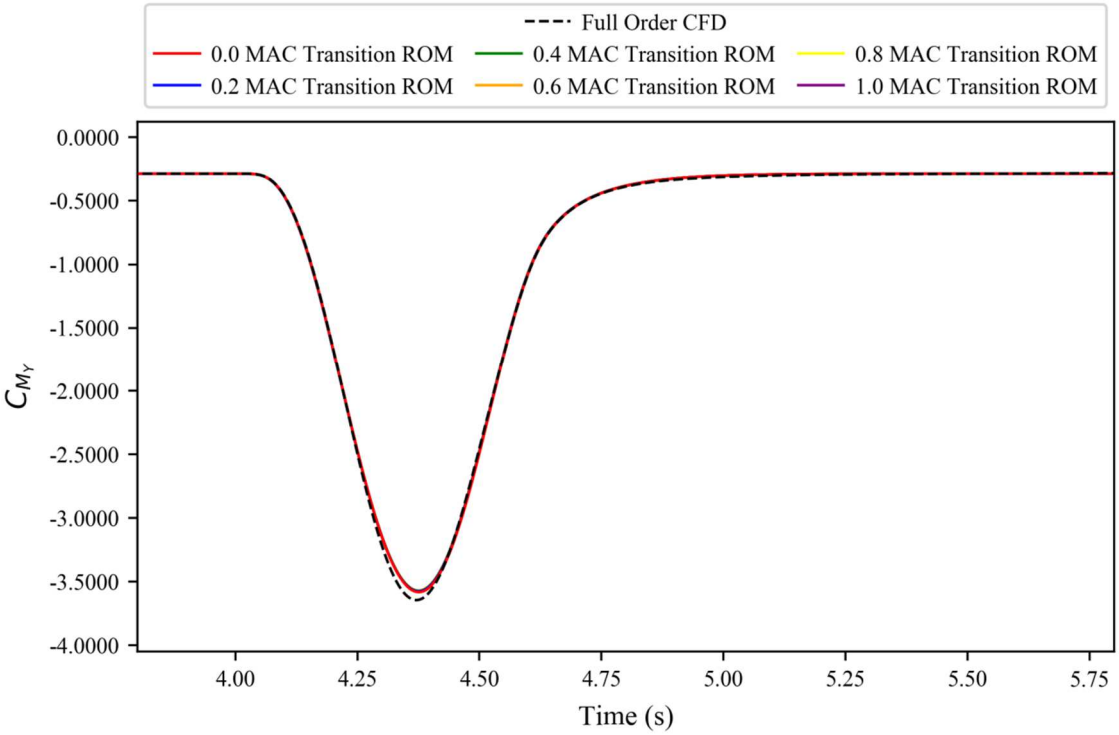

Figure 4.29 Variable time-step based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

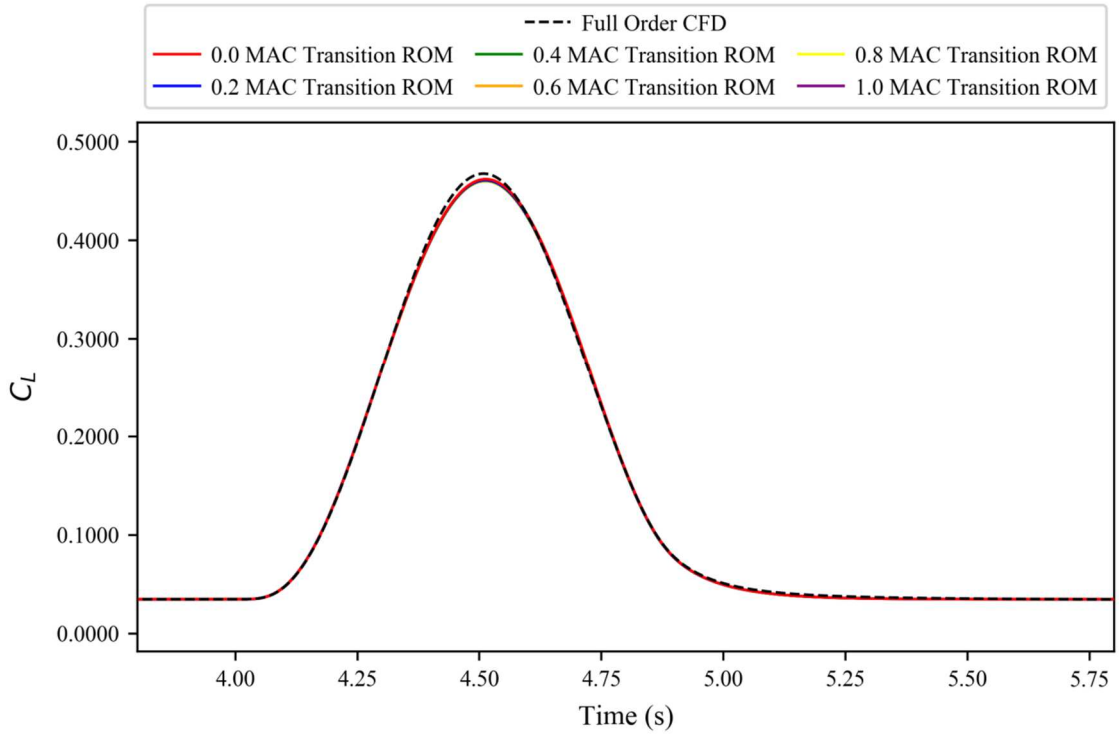

Figure 4.30 Variable time-step based ROM method results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

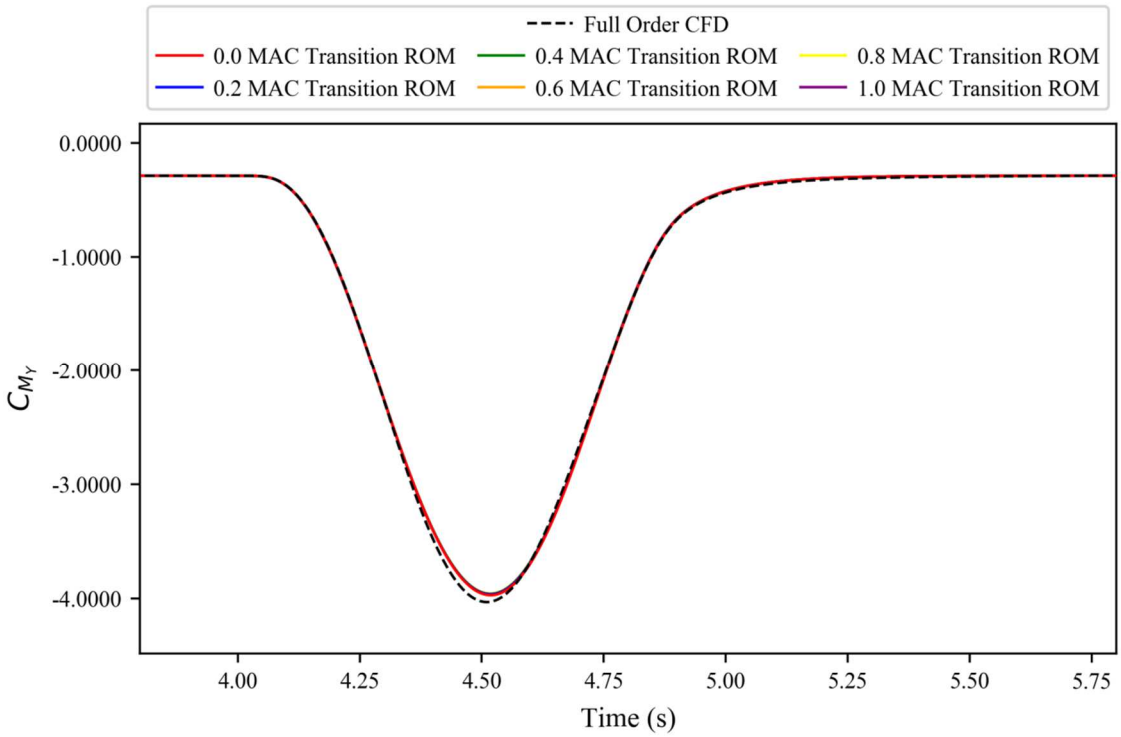

Figure 4.31 Variable time-step based ROM method results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

As the accuracy is so high, it is necessary to examine the peak responses in more detail to see any difference in the ROM response caused by the various transition points. Figures 4.32-4.39 zoom in on the peaks shown in Figures 4.24-4.31. Looking at the peak results it can be seen that the transition point has minimal impact on the accuracy of the ROM. It can also be seen that the ROMs built with the sharp-edged gusts which transition between 0.2 and 1.0 MACs (ahead of the leading edge) all produce very similar results. The ROM built using the sharp-edged gust which transitions on the leading edge itself, however, can be seen to be something of an outlier; although one that still produces very similar results.

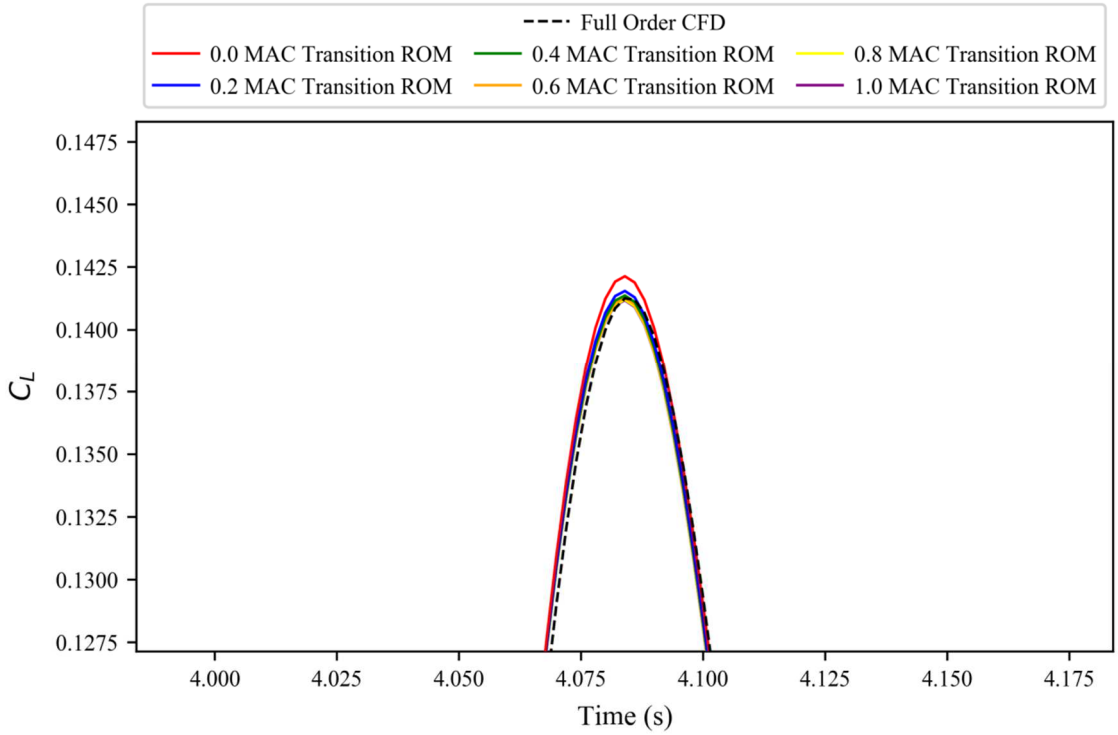

Figure 4.32 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

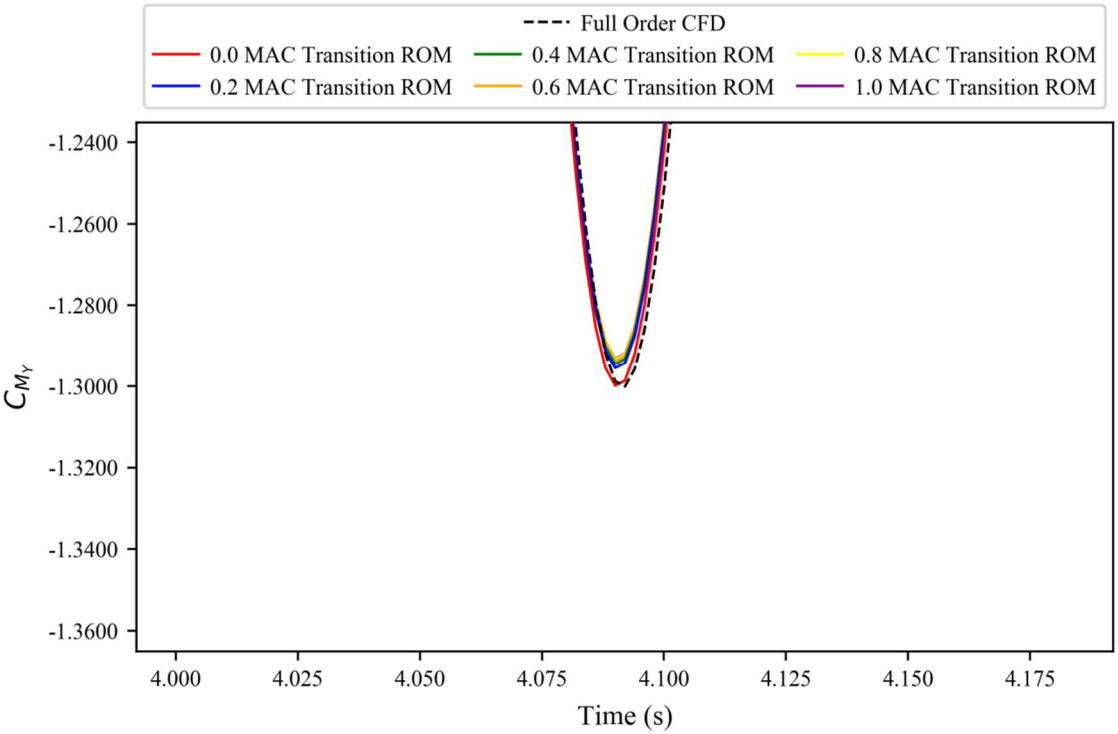

Figure 4.33 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

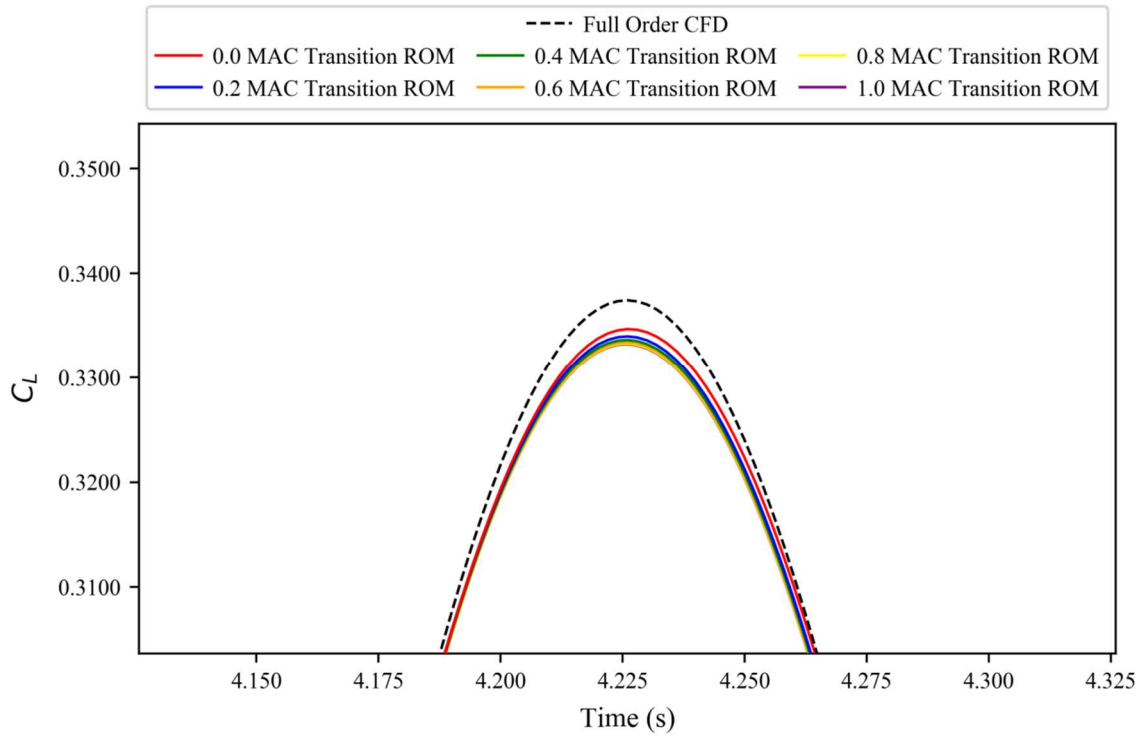

Figure 4.34 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

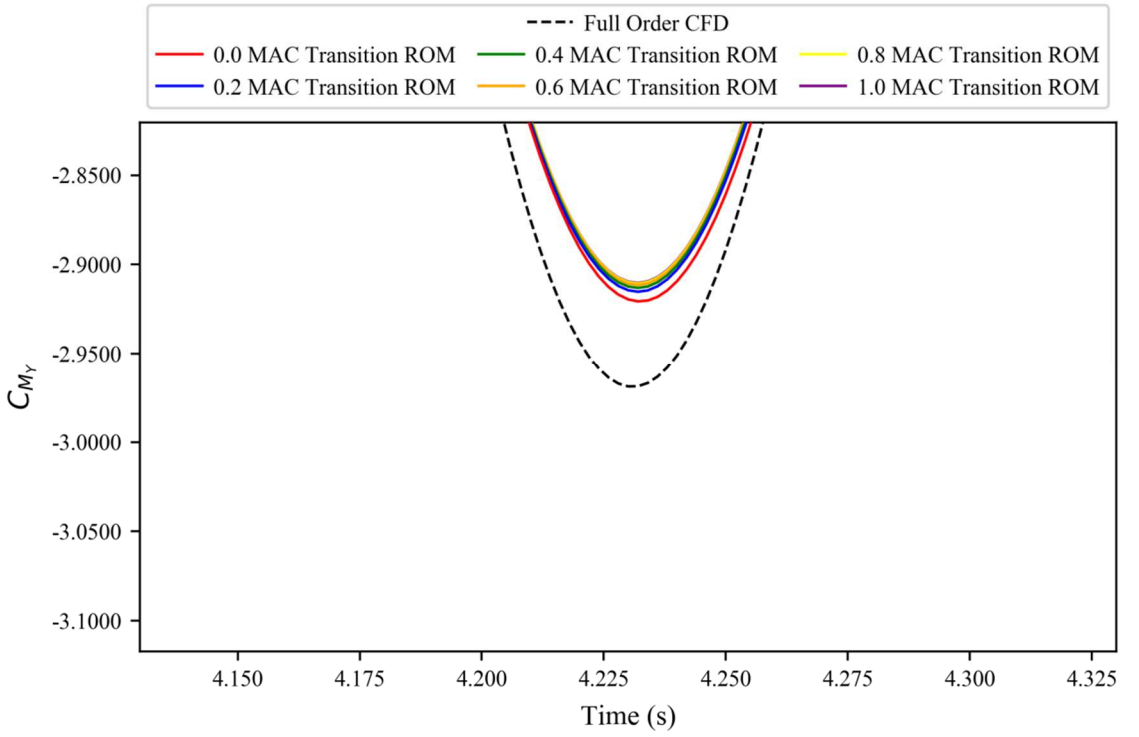

Figure 4.35 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

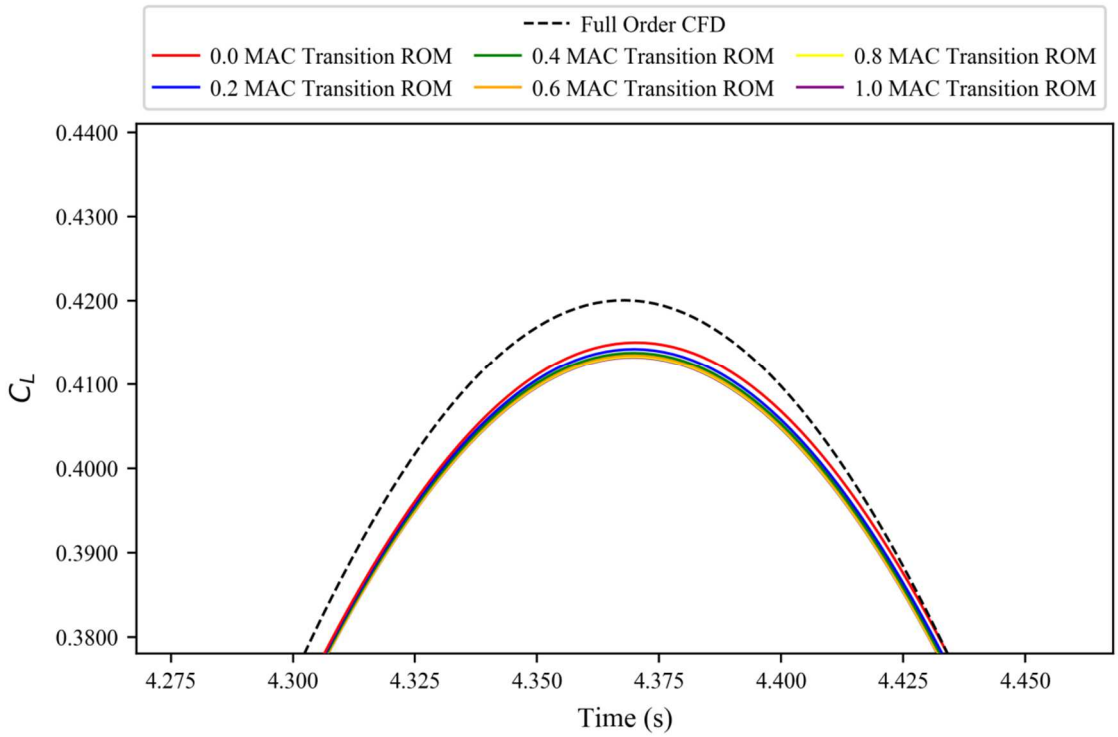

Figure 4.36 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

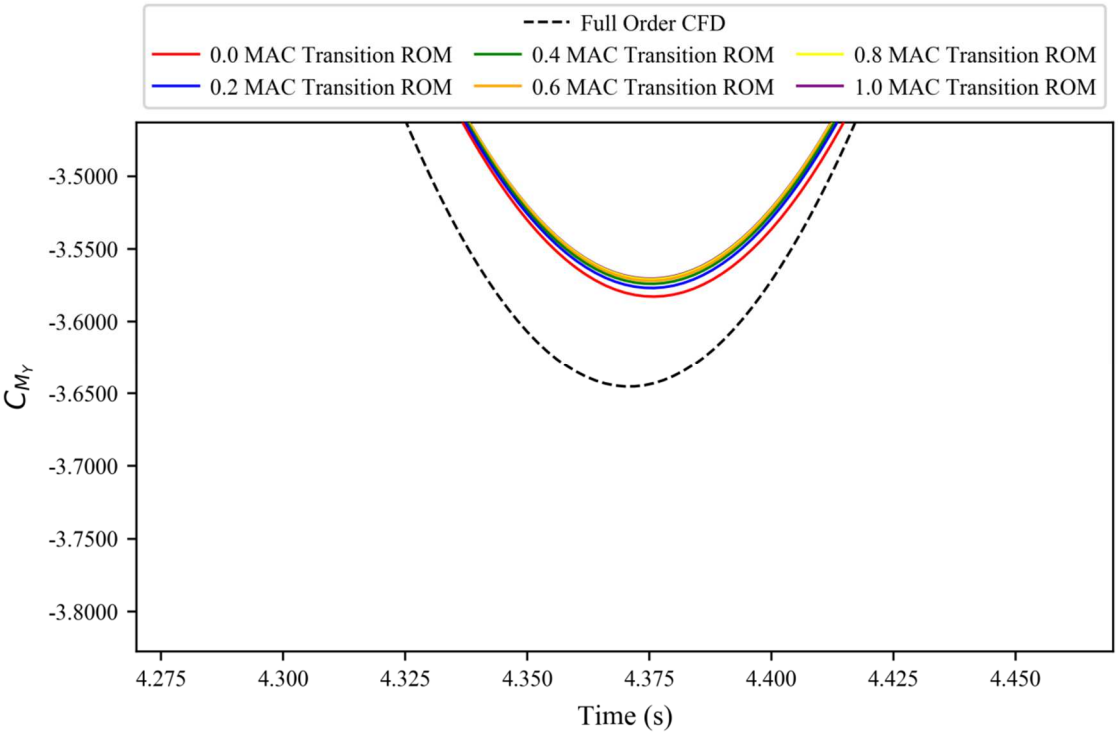

Figure 4.37 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

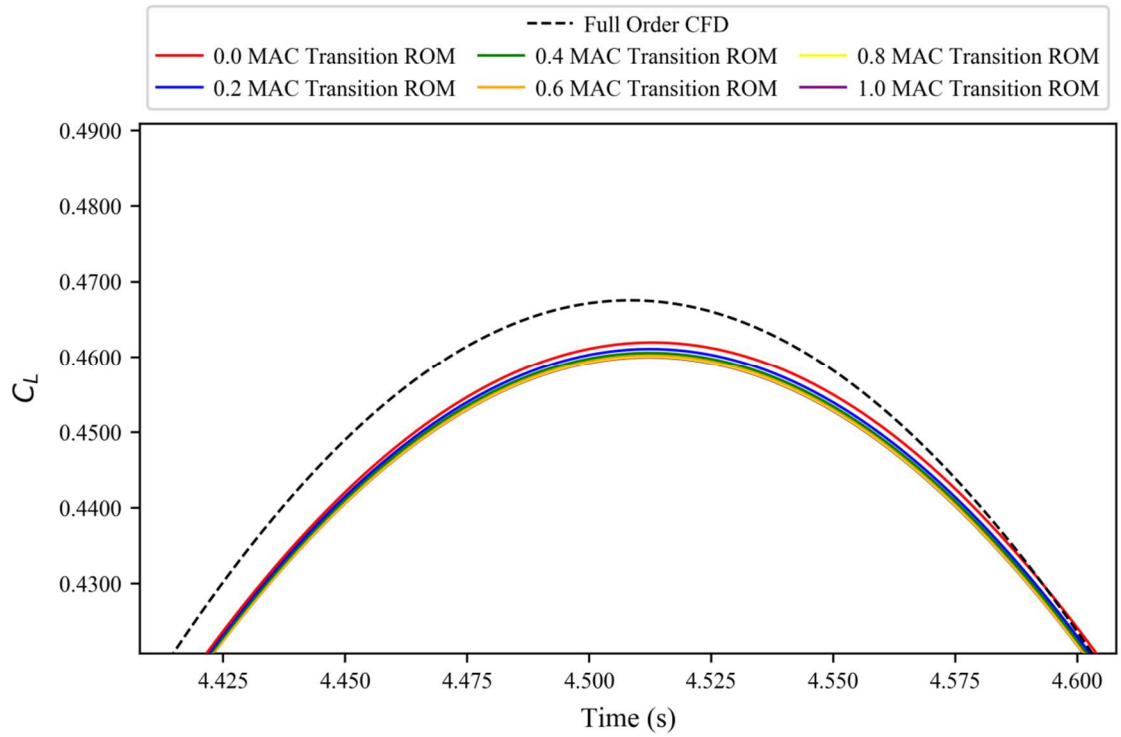

Figure 4.38 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

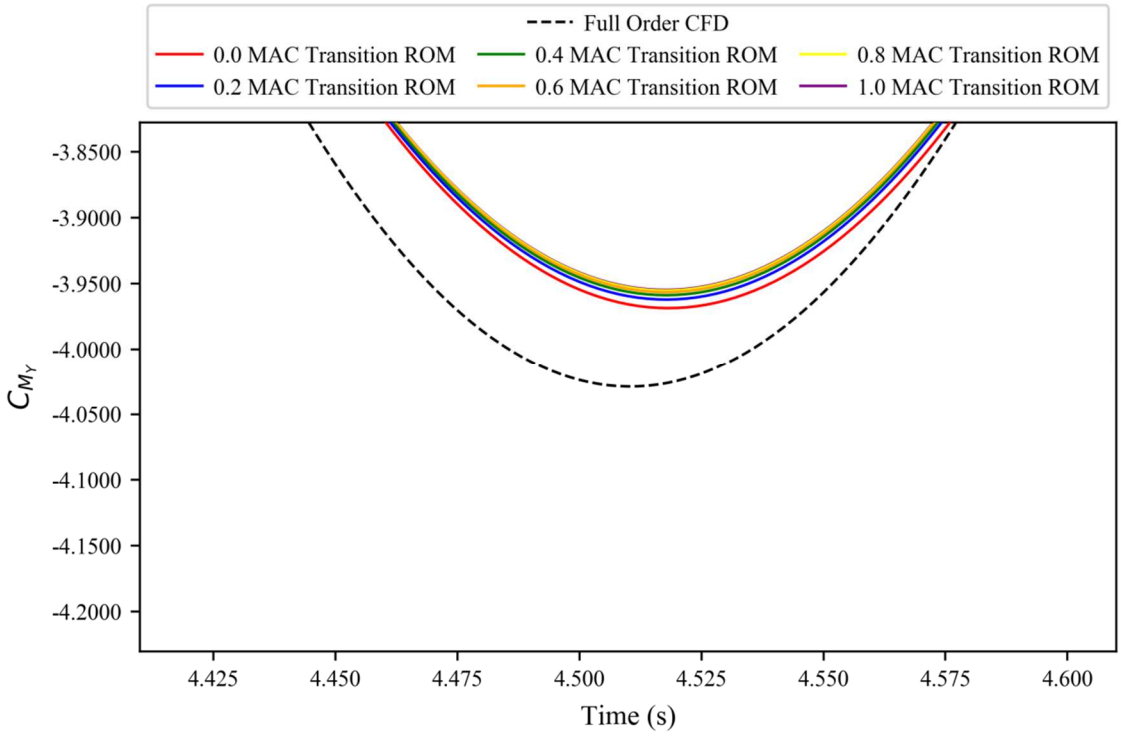

Figure 4.39 Peak response for a variable time-step based ROM against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

Whilst the full order CFD results do provide a guide as to the level of accuracy for a given ROM, it cannot be the only consideration. By considering Figures 4.20-4.39 it can be noted that as the transition point moves away from the leading edge, it is apparent

that the leading edge transition case coincidentally produces an error margin (compared to the other ROMs) that cancels out some of the error compared to the full order CFD. Crucially, this coincidence cannot be assumed to always produce advantageous results. As a result, and taking into account the fact that the effects of viscosity might increase the importance of the flow just ahead of the leading edge itself, it is logical (if somewhat counter intuitive based on the results) to view the 0.2 MAC transition based ROM is the best version to take forward as it appears to be the shortest distance ahead of the leading edge that transition can occur whilst not having any notable effect on the ROM output.

In terms of computational savings that variable time steps produces, it is hard to quantify as it is heavily dependent on the case setup (size of the domain, time step size, freestream velocity, etc.). However, a rough idea can be gained by examining this case. The distance between the domain and the start of the model was approximately 225 metres, the freestream velocity was roughly 250 metres per second, the small time step size was 0.002 seconds, the MAC length was approximately 7.7 metres and 4 large time steps were used. Based on these numbers, to start the sharp-edged gust outside the domain, it would require around 450 time steps to reach the model. For the 0.2 MAC transition case, the transition would occur around 3 small time steps before the leading edge of the model; with 4 large time steps, each around 0.2235 seconds long, being used to propagate the gust from outside the domain to the transition point. Based on these figures, transitioning from large to small time steps 0.2 MAC lengths ahead of the leading edge would result in a ~98% reduction in computational cost to propagate the sharp-edged gust from outside the domain to the model.

4.4. Length of Simulations

Having reduced the computational cost of the sharp-edged gust pre-impact, the post-impact aspect was considered; specifically, what was the minimum number of time steps required to accurately capture the system behaviour.

It was expected that most of the system response would be captured by the time the leading portion of the gust has gone past the aft most point of the model. However, it was also deemed possible that some aspects of the response would not be captured until the gust had travelled further downstream and the system had had time to start settling.

Building upon the ROM method from Section 4.3.2, a new set of ROMs were constructed using a single sharp-edged gust that transitioned from large to small time steps 0.2 MAC lengths ahead of the leading edge. The data from this sharp-edged gust was then artificially cut when the gust reached different locations downstream; measured in model lengths (M.L.) post-impact (see Figure 4.1). These sharp-edged gust responses of various lengths were then used to construct the ROMs in the same manner as before; again with a ROM-size of 20.

4.4.1. Initial Results

Again, to test the new ROMs, the system responses to the four gust lengths of flight point two were simulated.

Figures 4.40-4.47 show an unexpected result in that far more data than expected was required to produce reasonable results. Indeed, the sharp-edged gust appears to require data for 7-9 model lengths post-impact in order for the results to converge; although it should also be noted that in some cases, particularly the smaller gust cases, some of the results are similar enough as to be hard to distinguish from each other on the plots.

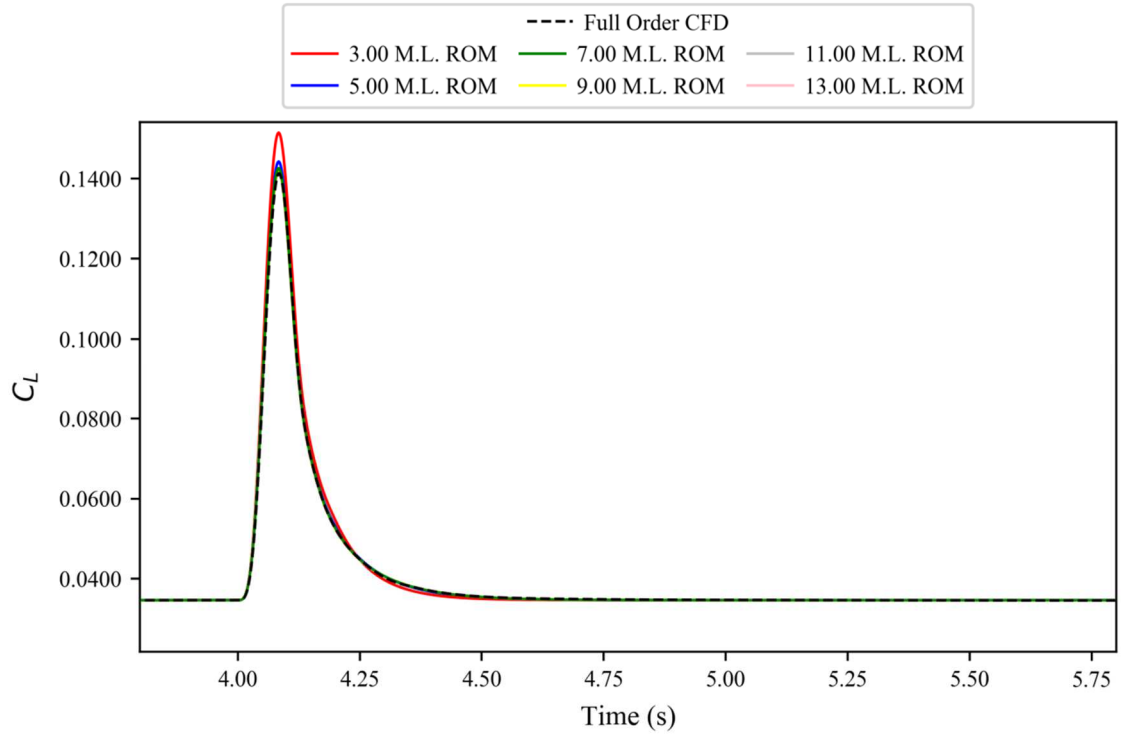

Figure 4.40 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

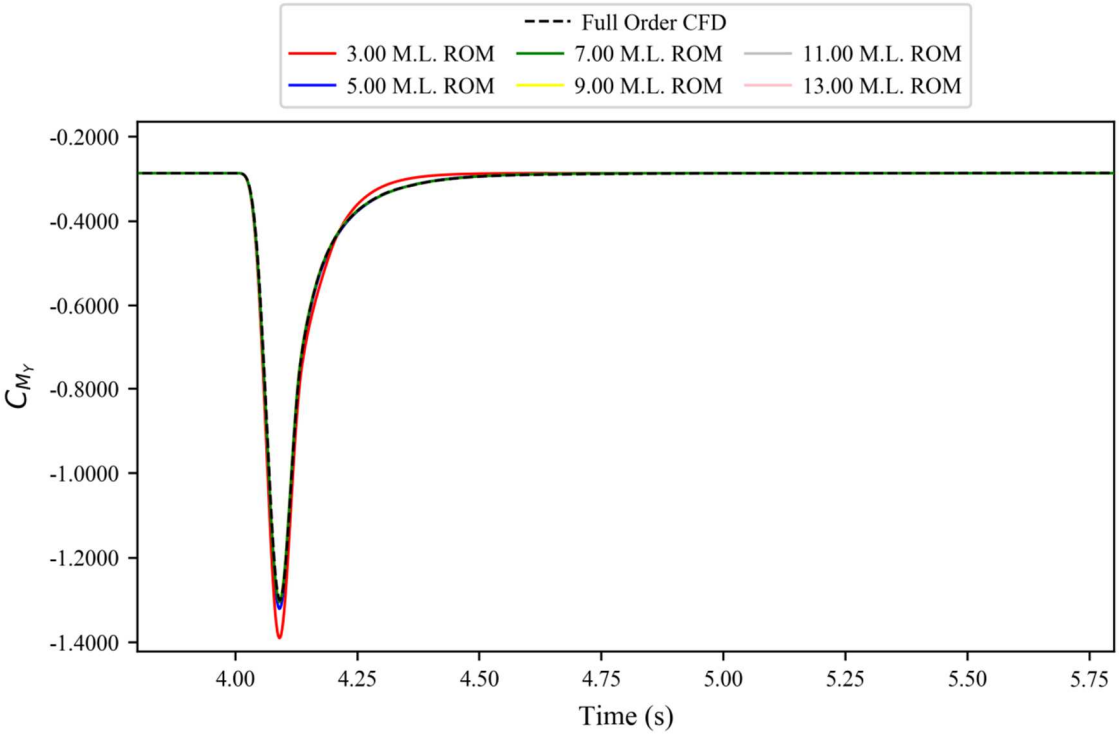

Figure 4.41 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

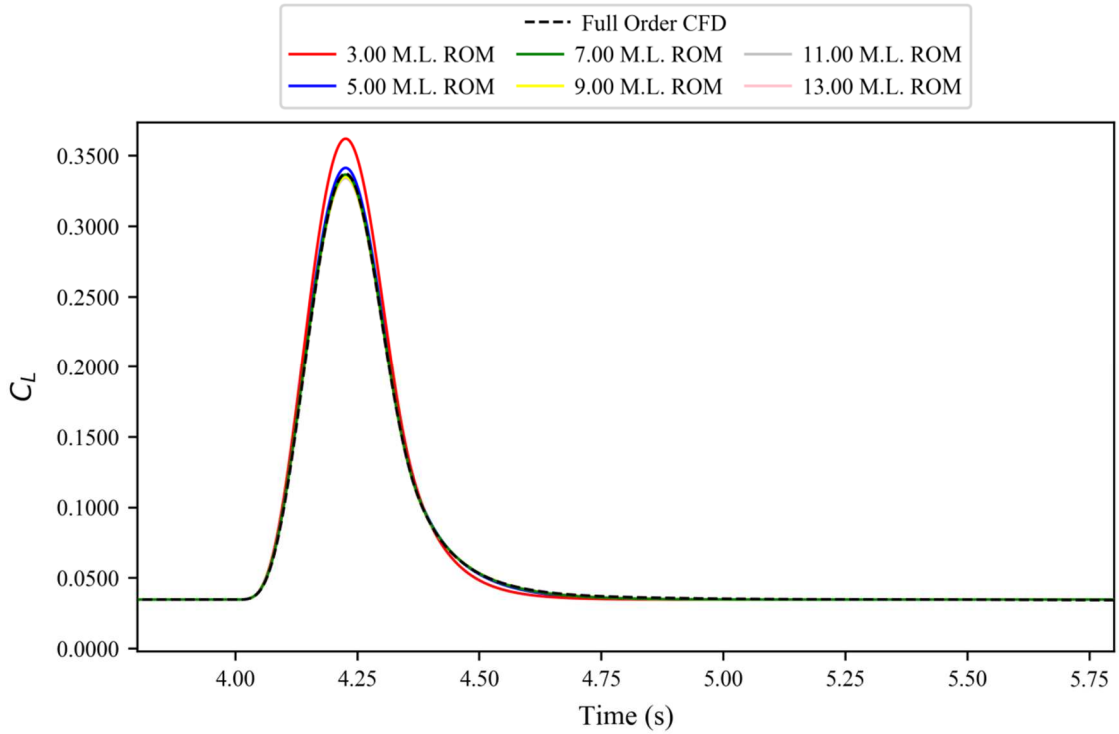

Figure 4.42 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

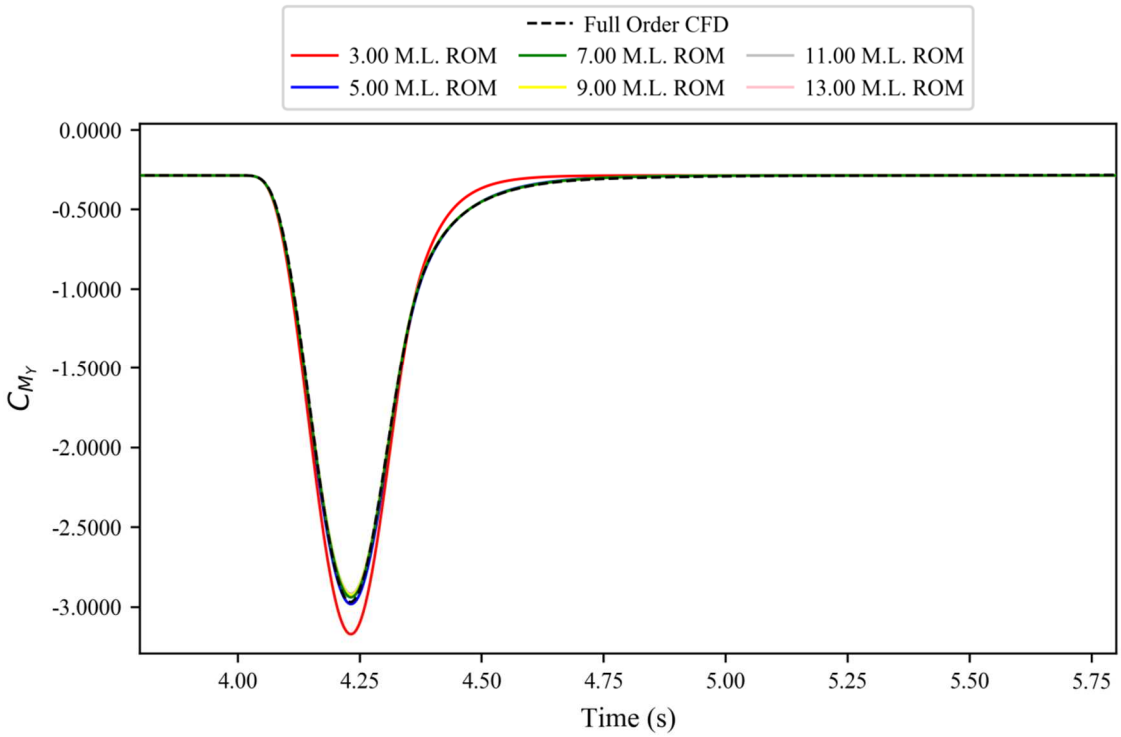

Figure 4.43 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

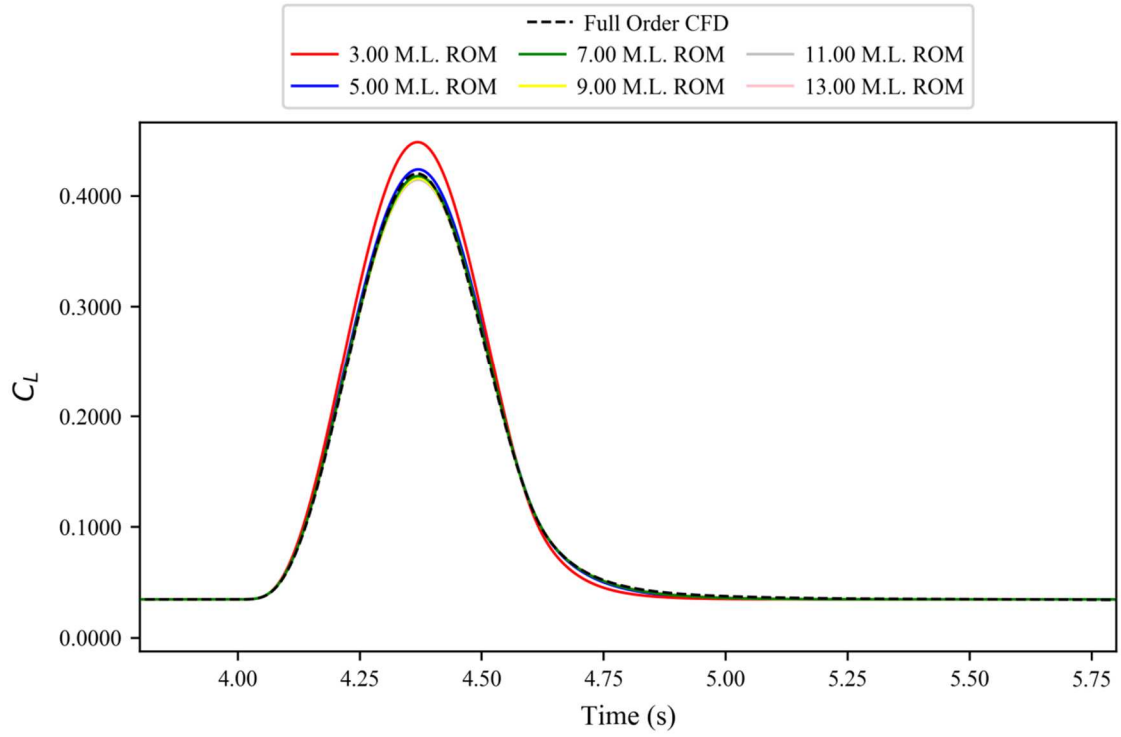

Figure 4.44 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

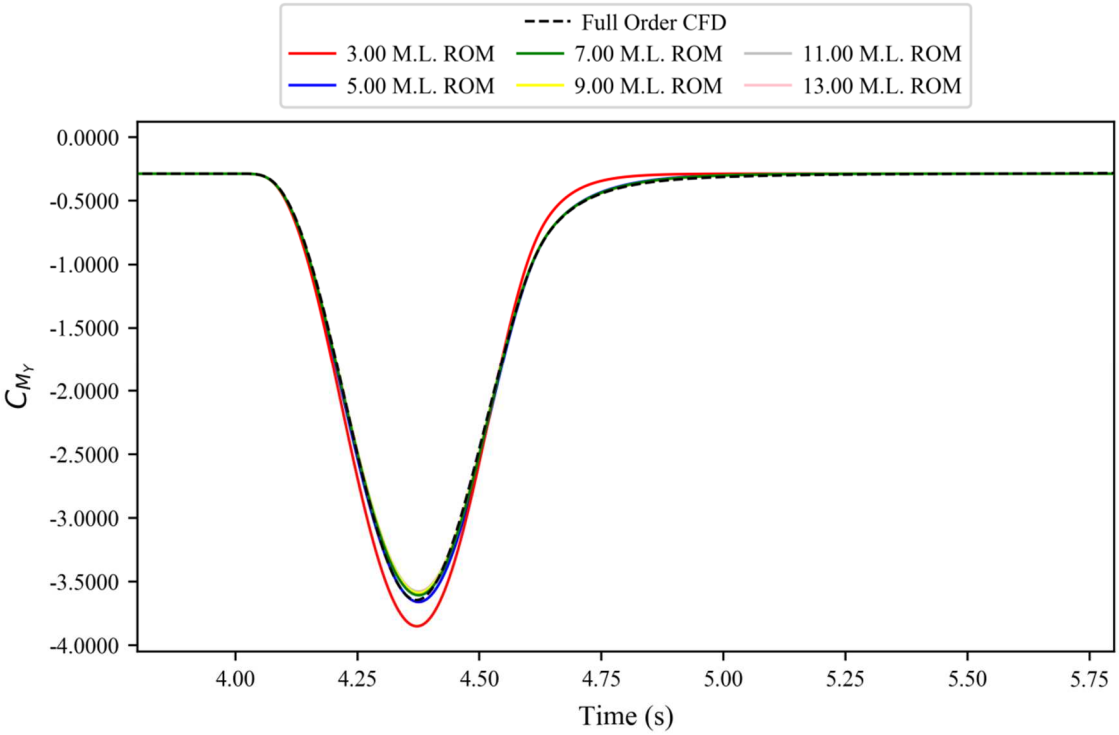

Figure 4.45 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

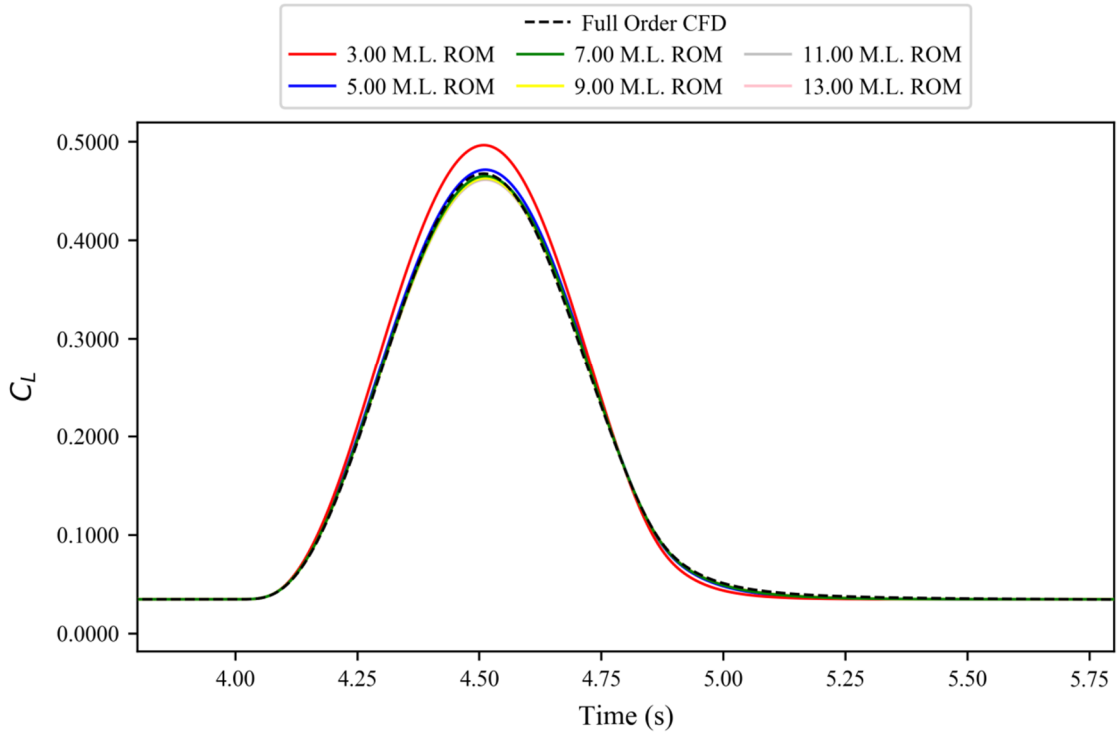

Figure 4.46 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

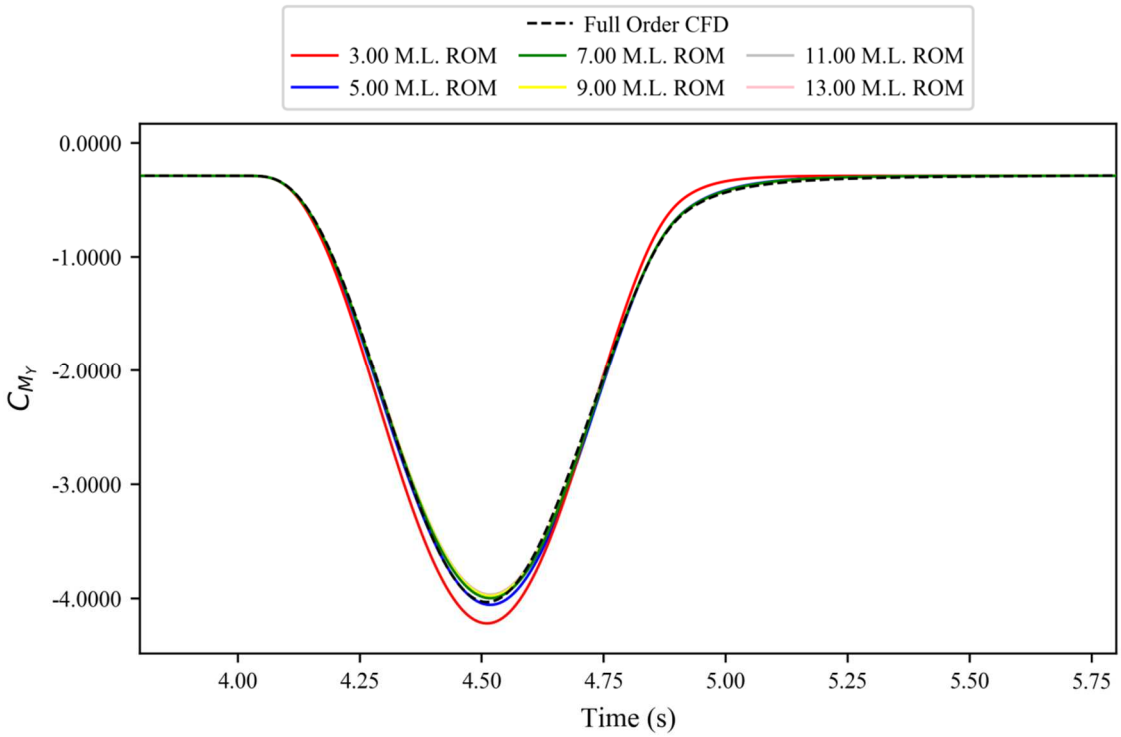

Figure 4.47 Short sharp-edged gust based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

4.4.2. Modified Linear Gradient Method

The results shown in Figures 4.40-4.47 were found to be due to the way in which the linear gradients (used within the steady state correction) were calculated. Previously, the 1ms^{-1} sharp-edged gust was used to calculate the linear gradients, taking advantage of how the difference in force coefficients before and after the gust impacted also happened to equal the linear gradients for these coefficients. However, when the sharp-edged gust is not run for long enough the system does not have time to settle, and so the final time step in the results will not correspond to the correct, settled values; causing the linear gradients calculated to degrade as the response is ended increasingly early.

In order to overcome this problem a new method of calculating these linear gradients was implemented. First, the polar sweep is carried out as normal; for each step, the simulation is setup so that the aerofoil is at an angle of attack (α) and the freestream velocity has no vertical component (see Figure 4.48). Next, each case is treated as if it has a zero angle of attack, but instead with a freestream velocity coming in at angle α (see Figure 4.49). Finally, this angled freestream velocity is broken down into its

component velocities to obtain an equivalent vertical gust velocity (see Figure 4.50); this is done by applying Eqn. (4.2):

$$v_g = v_\infty \sin(\alpha) \tag{4.2}$$

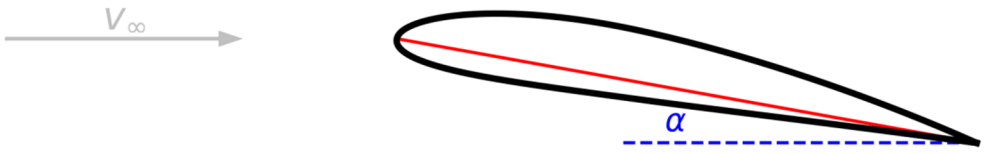

Figure 4.48 Actual setup for a polar sweep simulation at angle of attack α.

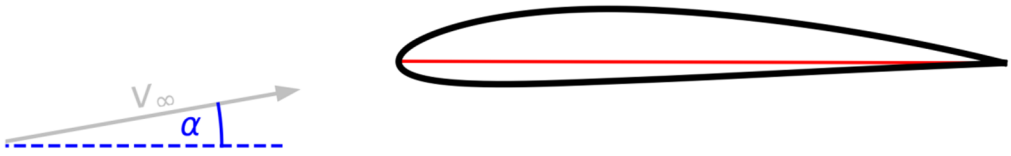

Figure 4.49 Equivalent zero angle of attack scenario for a polar sweep simulation at angle of attack α.

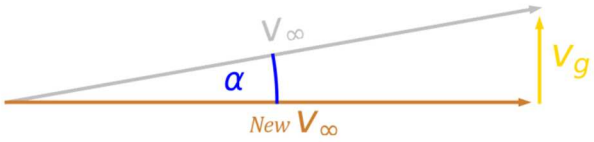

Figure 4.50 Component velocities (including equivalent vertical gust velocity) for an equivalent zero angle of attack scenario for a polar sweep simulation at angle of attack α.

Once this series of calculations have been completed, two different angles of attack are selected (typically 0 and 0.5 degrees) the linear gradients can then be calculated similarly before using these two equivalent gust results (see Section 3.3).

4.4.3. Modified Linear Gradient Results

Applying the new method for calculating the linear gradient, the impact it has can be seen when calculating the system response of the FFAST wing for the four gusts of

flight point 2. It is worth highlighting that as the modified linear gradient method improves the results considerably, the amount of data needed can be decreased substantially from those needed in Section 4.4.1. As such, the plots look worse at a glance, however this is due to the largest amount of data being one 3 model lengths of post-impact time steps; whereas in Section 4.4.1 this was the least amount of data used.

Figures 4.51-4.58 show that ending the sharp-edged gust 1.33 model lengths post-impact does not appear to be sufficient; producing an unstable result that manifests as a high inaccurate result, or else one which shows oscillation and in some cases, exponential growth after initially appearing to have been damped out. The ROMs built with the 2.00 model lengths post-impact sharp-edged gust is the first case which produced reasonable results, however even then it does not really capture the settling of the system as well as desired. The capture of the settling of the system does improve slightly as the sharp-edged gust is run increasing further downstream, however even the 3.00 model length ROM does not really capture the settling to a satisfactory level.

Again, it is should be noted that in some cases the results can be hard to distinguish from each other as they are similar enough as to effectively exist on top of each other for much of the response. Additionally, any instance where the response was unstable (see Sections 1.5.5 and 3.4) and growing have been ended early to make the plot easier to read.

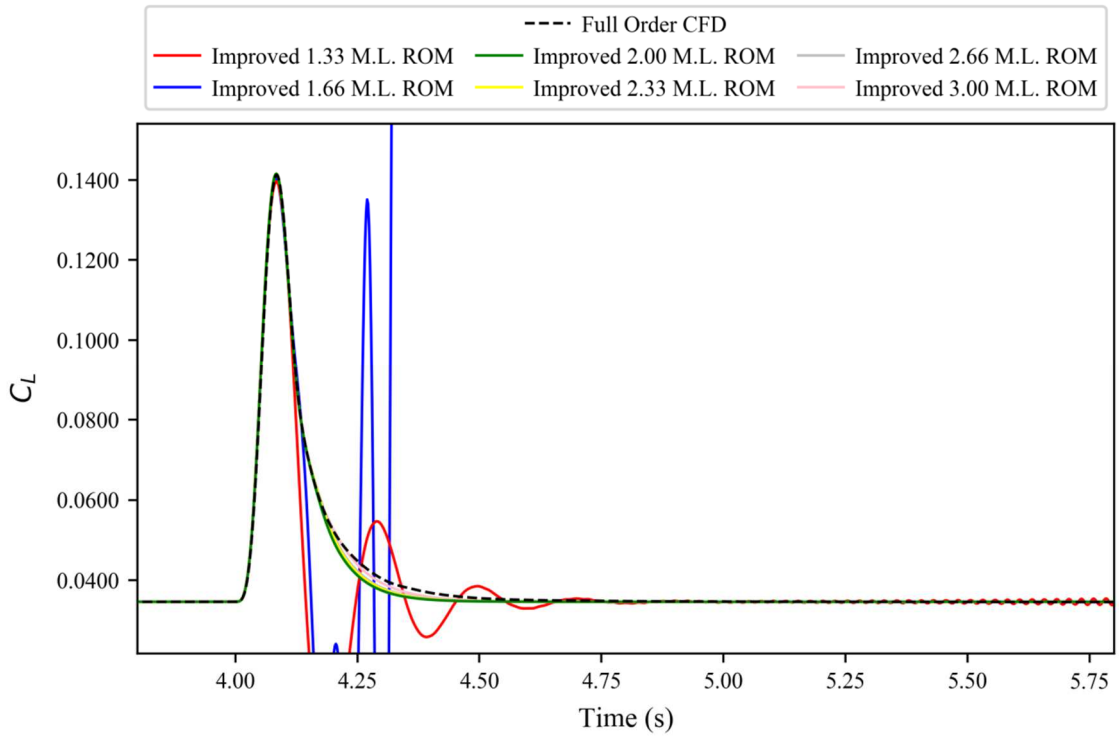

Figure 4.51 Modified linear gradient ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

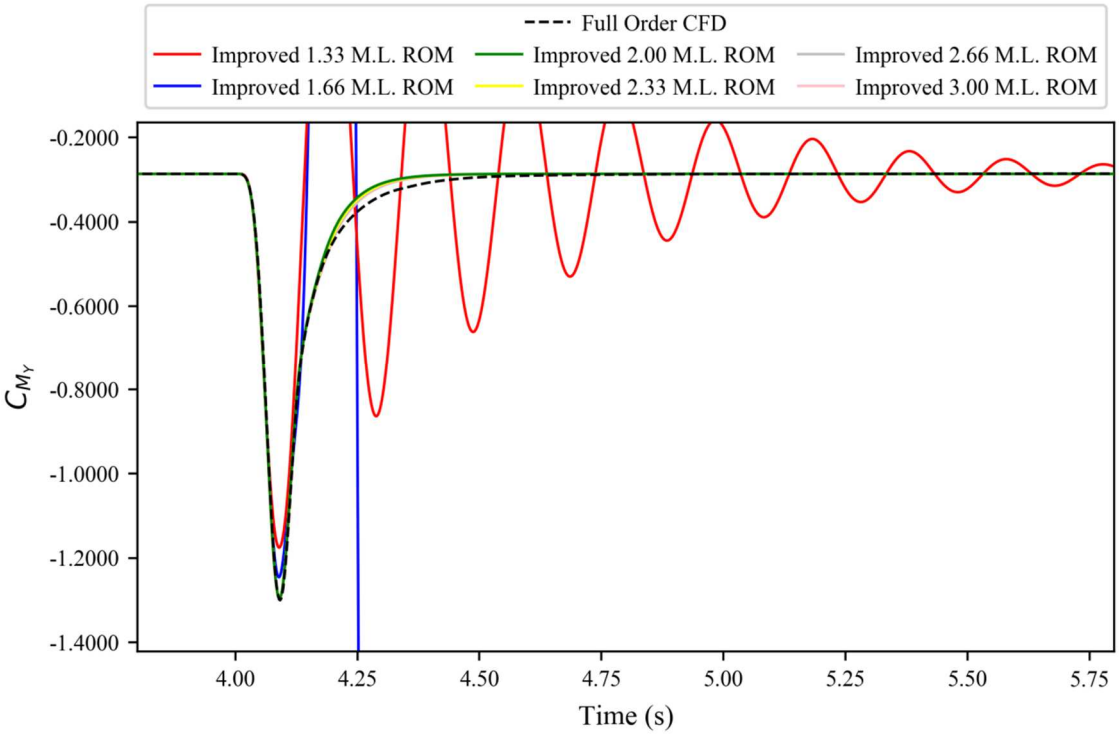

Figure 4.52 Modified linear gradient ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

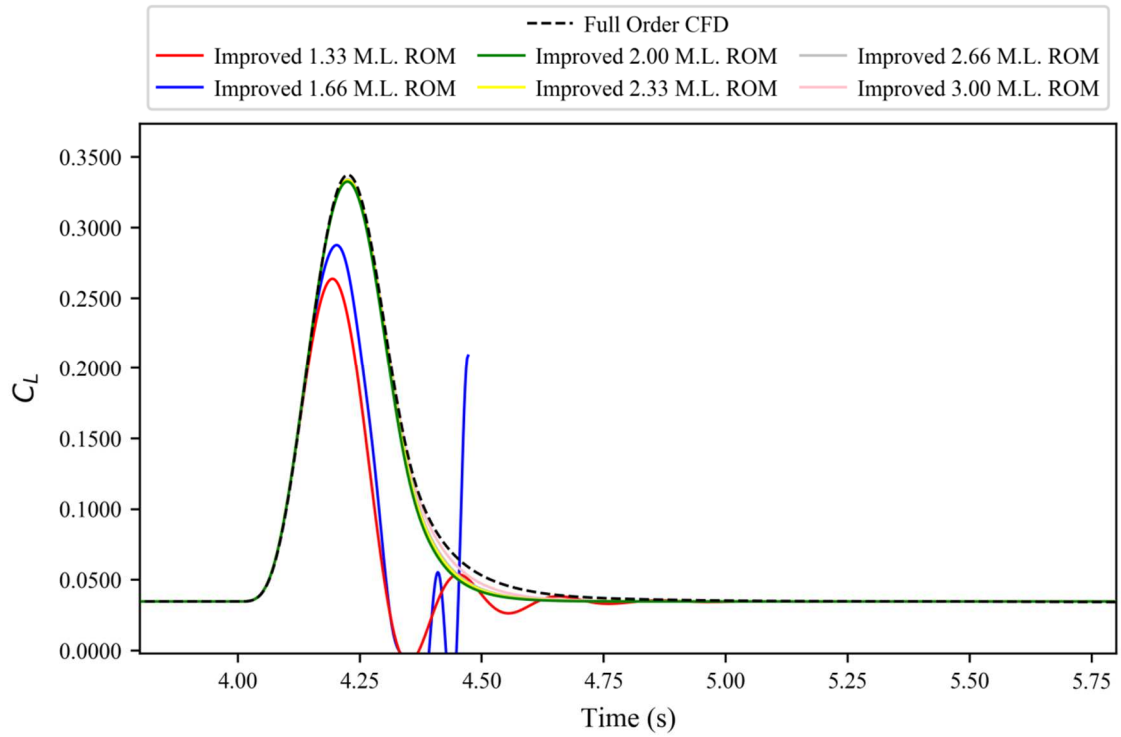

Figure 4.53 Modified linear gradient ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

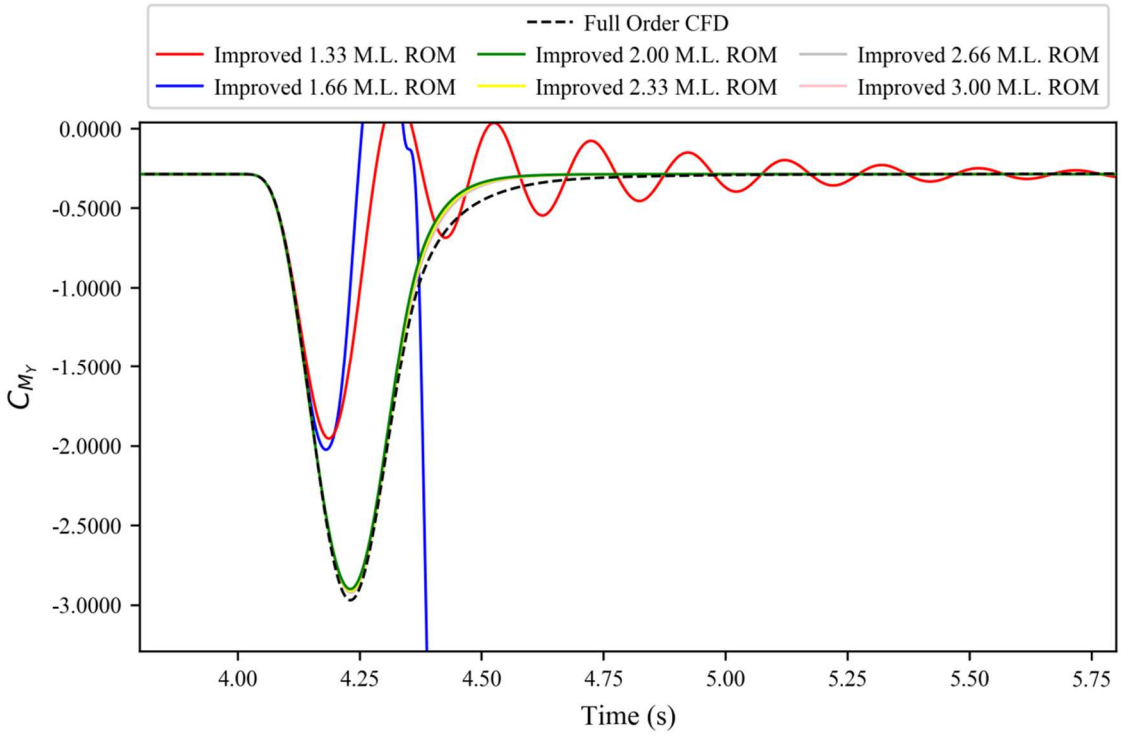

Figure 4.54 Modified linear gradient ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

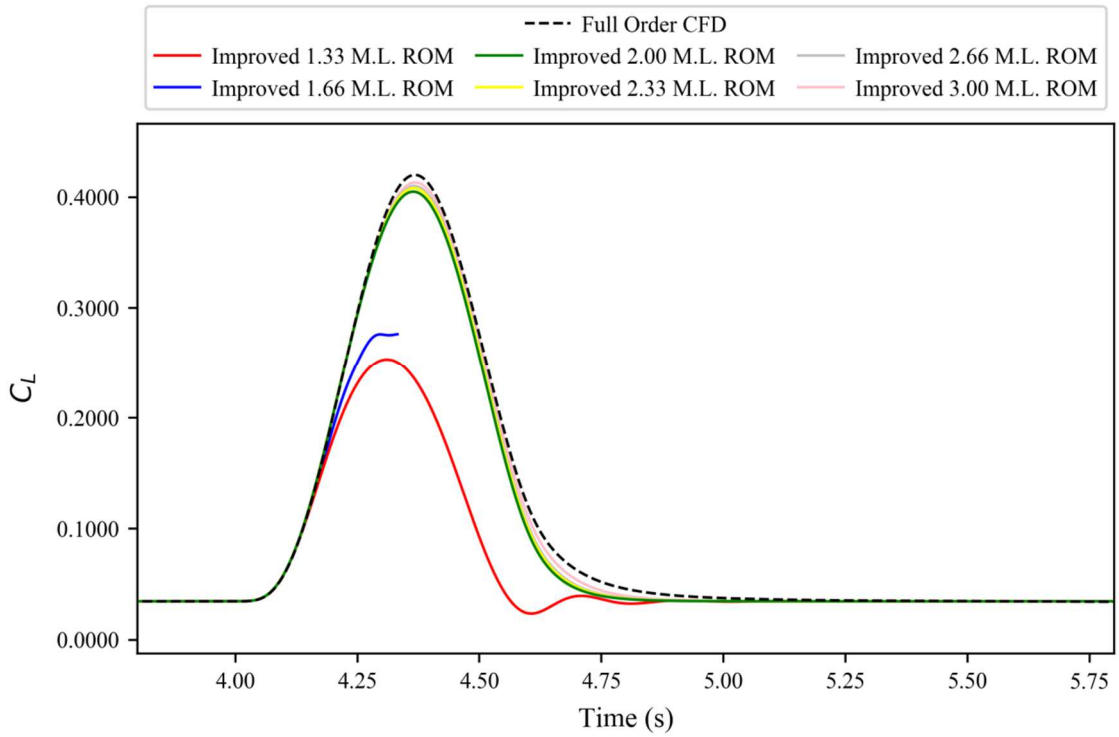

Figure 4.55 Modified linear gradient ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

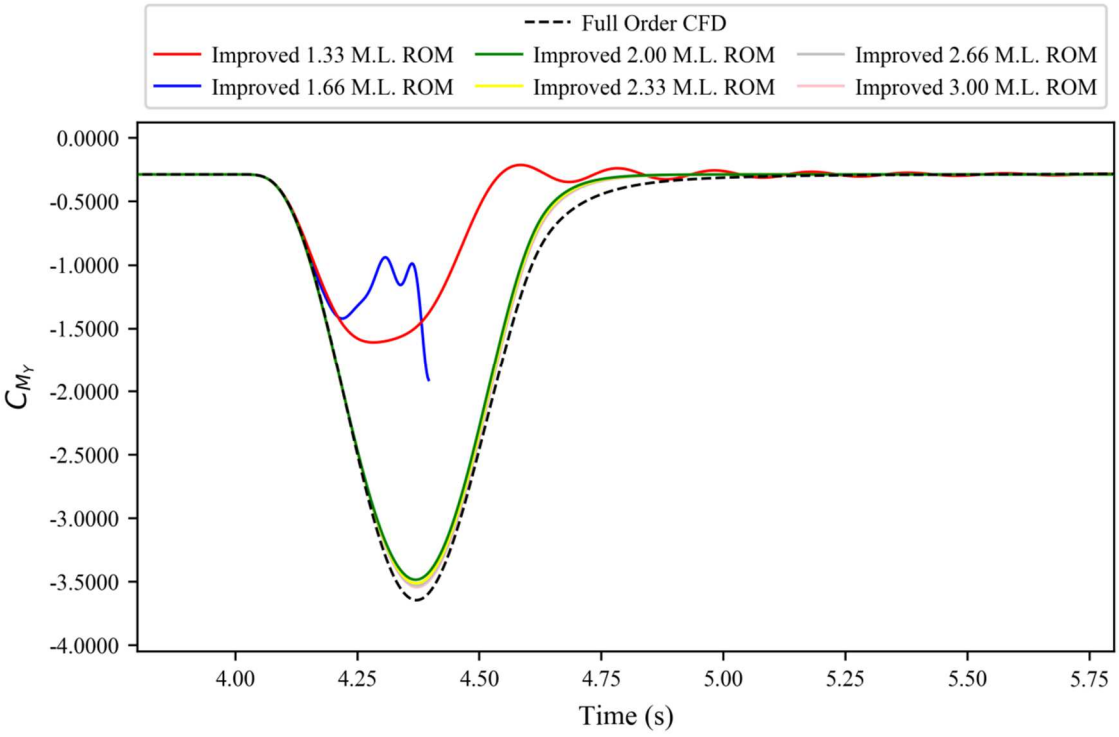

Figure 4.56 Modified linear gradient ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

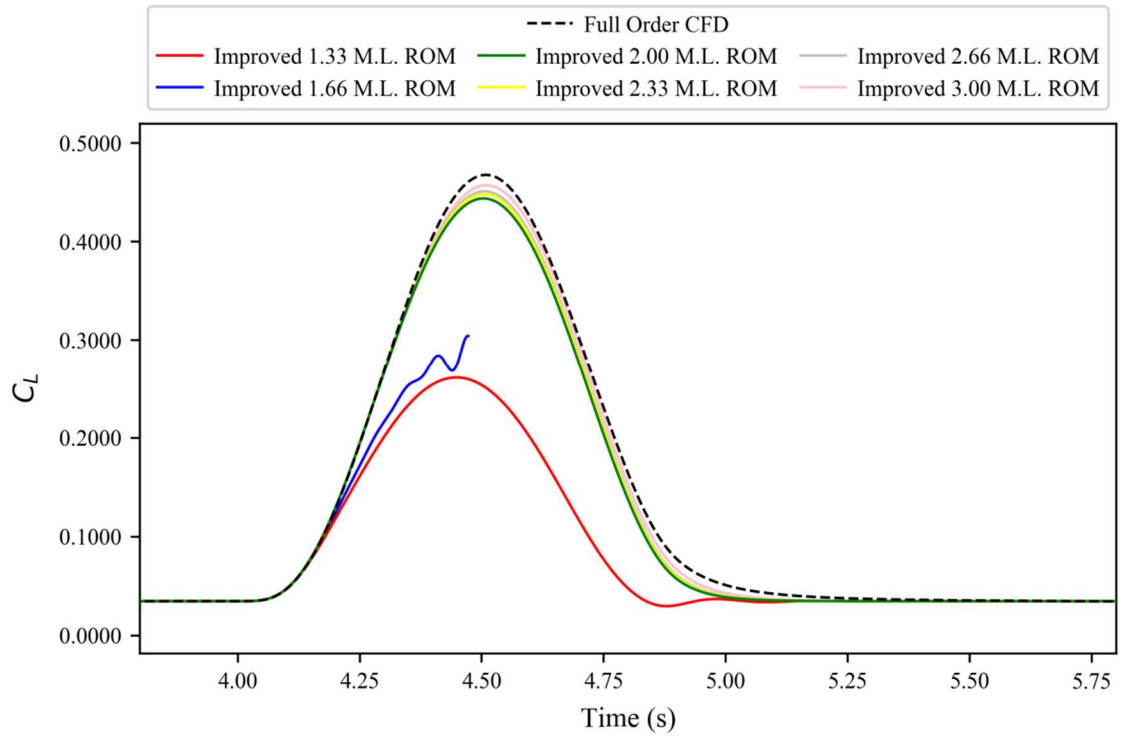

Figure 4.57 Modified linear gradient ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

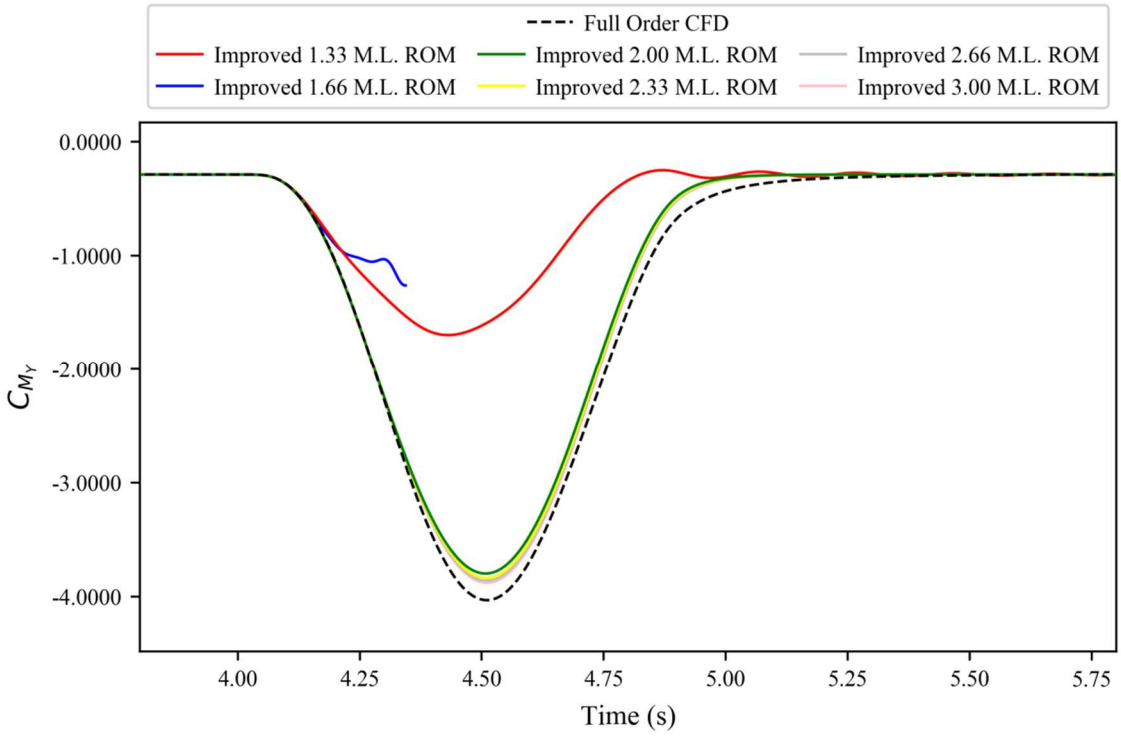

Figure 4.58 Modified linear gradient ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

4.4.4. Step-down ROM Method

To improve the settling of the system without running the sharp-edged gust for considerably longer than 3.00 post-impact model lengths, it was also found to be necessary to modify the way in which the Hankel matrix is constructed. For all ROMs presented to this point, the sharp-edged gust is used to calculate the pulse response of the system, which is in turn used to construct the Hankel matrix. However, if the sharp-edged gust is subtracted from the steady state of the system an effective step-down response can be calculated. Instead of then calculating the pulse response of the system, this step-down response can be used directly to construct the Hankel matrix as laid out in Section 3.2.4. This output response can be adjusted to occur at the correct baseline. Ultimately, as the first term of the new Hankel matrix is the equivalent of the steady state of the system, it helps to capture the settling of the system to this state in a more accurate manner, even with less data available.

4.4.5. Step-down ROM Results

Applying this new method of constructing the Hankel matrix, and still using a ROM-size of 20, the system response of the FFAST wing for the four gusts cases of flight point two can be simulated.

Figures 4.61-4.66 show that the new step-down based ROM method captures the settling of the system much better than the old pulse based ROM method (Figures 4.51-4.58); apart from the 1.33 model length case. The ROM built with the 1.33 model length sharp-edged gust is highly unstable, but the ROMs built with 1.66 model length sharp-edged gusts start to capture the settling of the system as desired. However, the ROM built with the 1.66 model length sharp-edged gust produces some unexpected results just prior to the final settling of the system for the first gust length (Figures 4.59-4.60). Therefore, based on the settling of the system, the ROMs built from the sharp-edged gust with either 2.00 or 2.33 model lengths post-impact data is considered the best in terms of balancing computational cost with maintaining a high level of accuracy.

As before, results that are very similar can be hard to distinguish from one another on the plot for much of the response. Additionally, unstable responses (see Sections 1.5.5 and 3.4) which are growing have been ended early to make the plot easier to read.

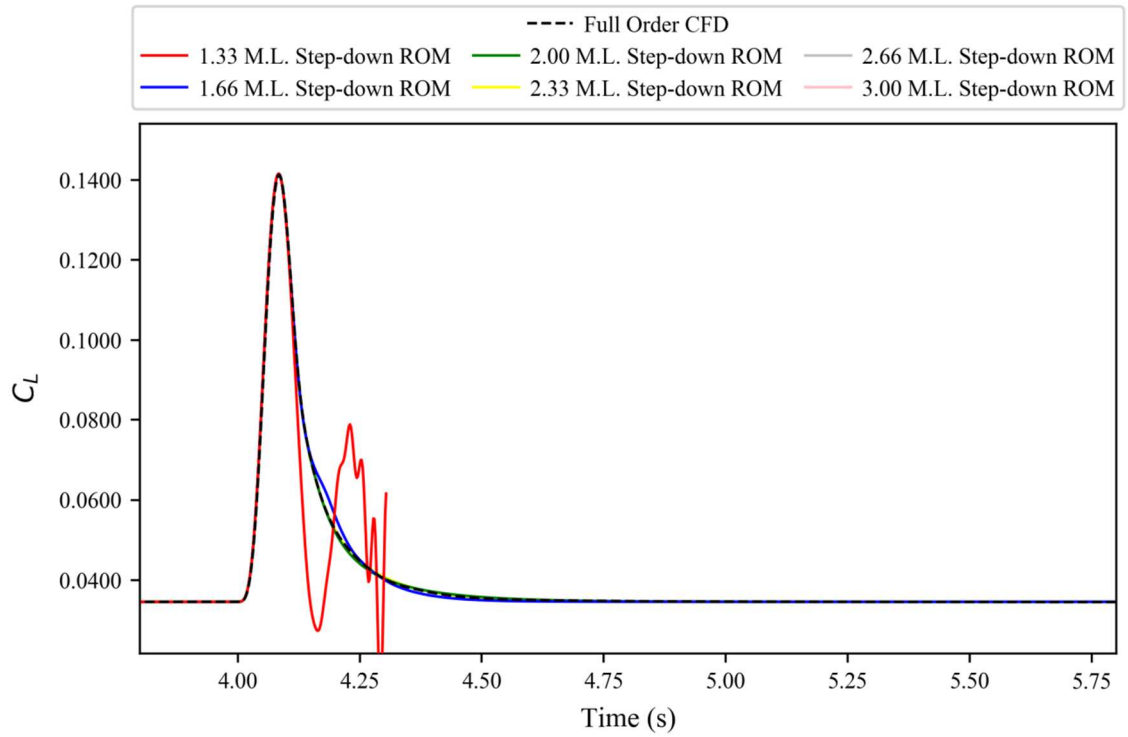

Figure 4.59 Step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

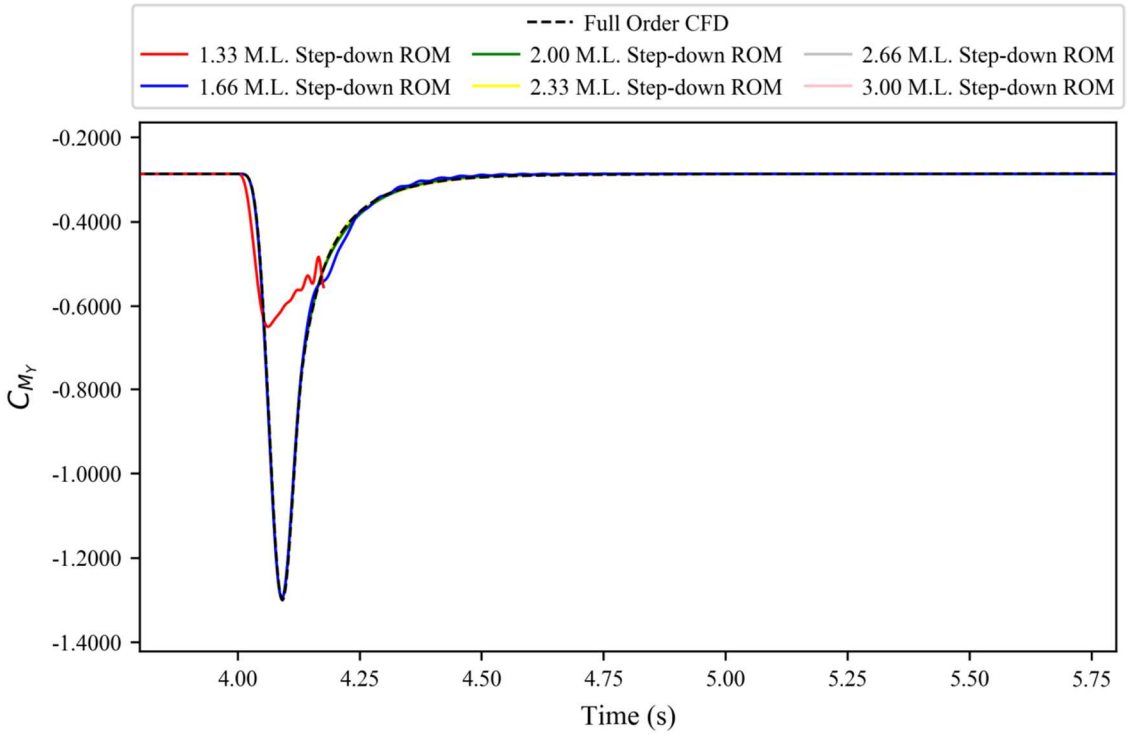

Figure 4.60 Step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

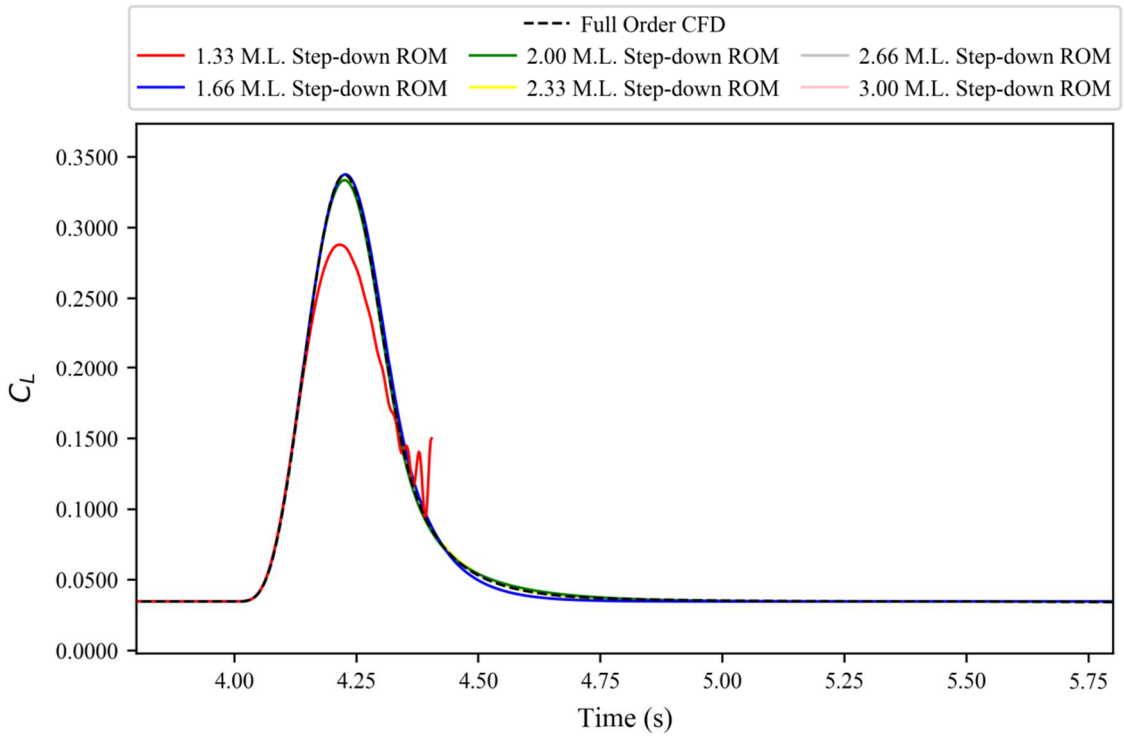

Figure 4.61 Step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

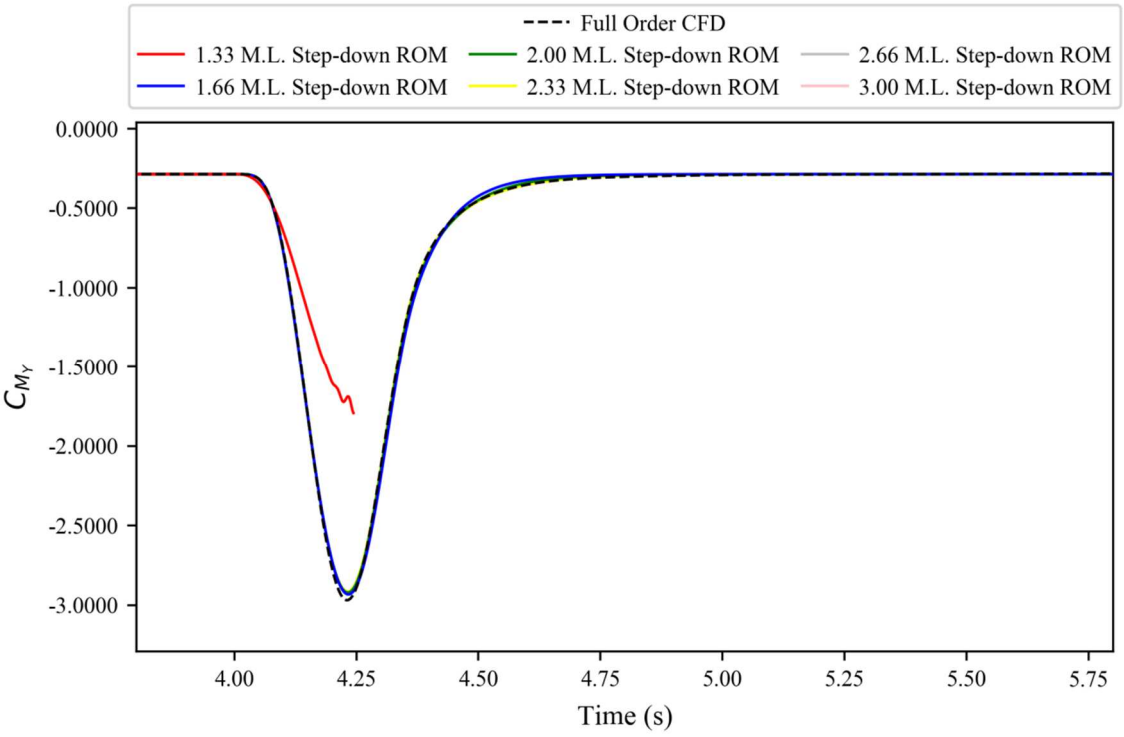

Figure 4.62 Step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

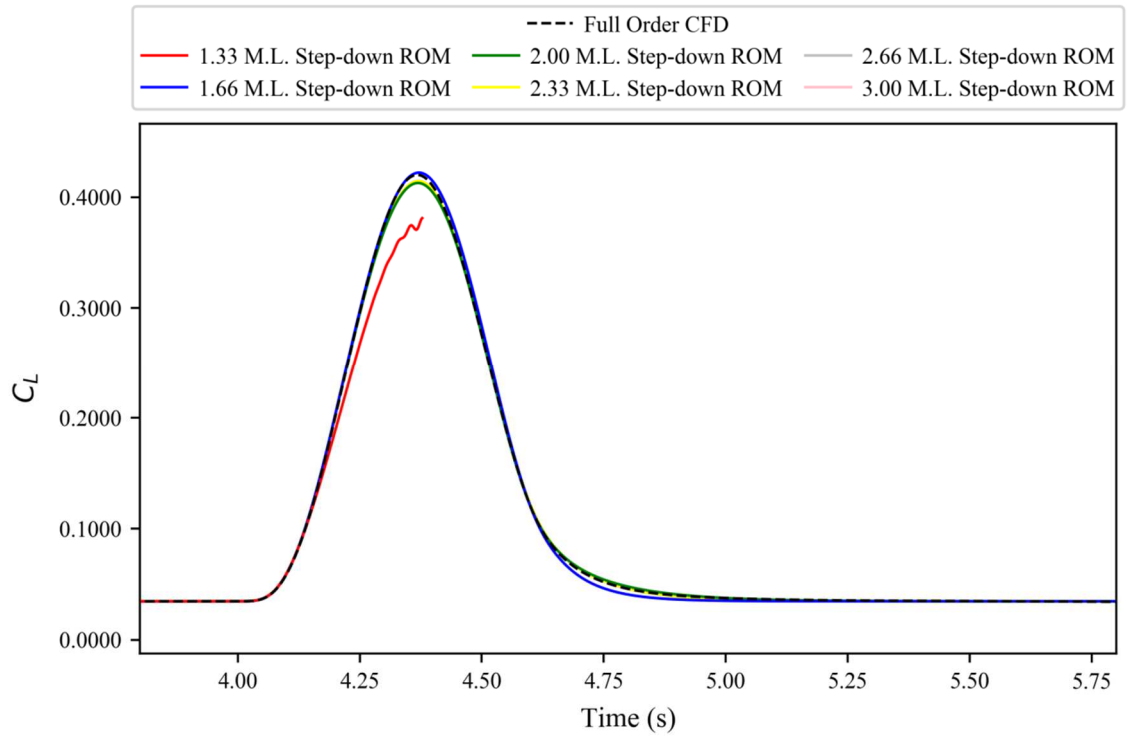

Figure 4.63 Step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

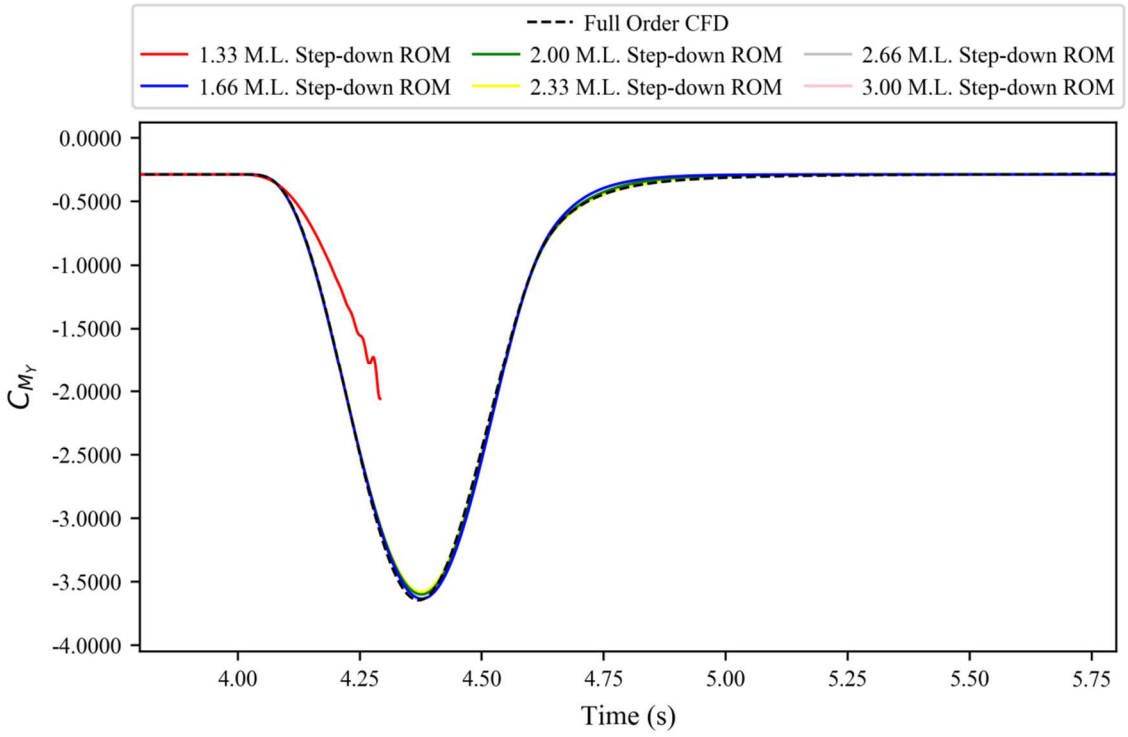

Figure 4.64 Step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

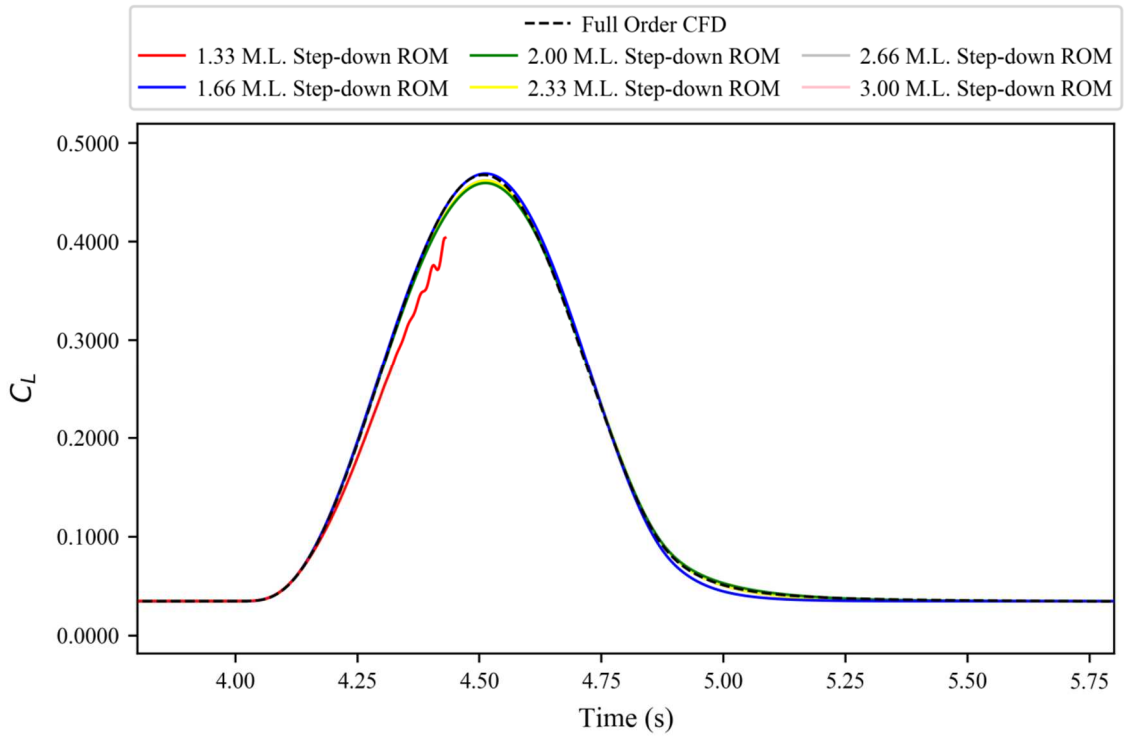

Figure 4.65 Step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

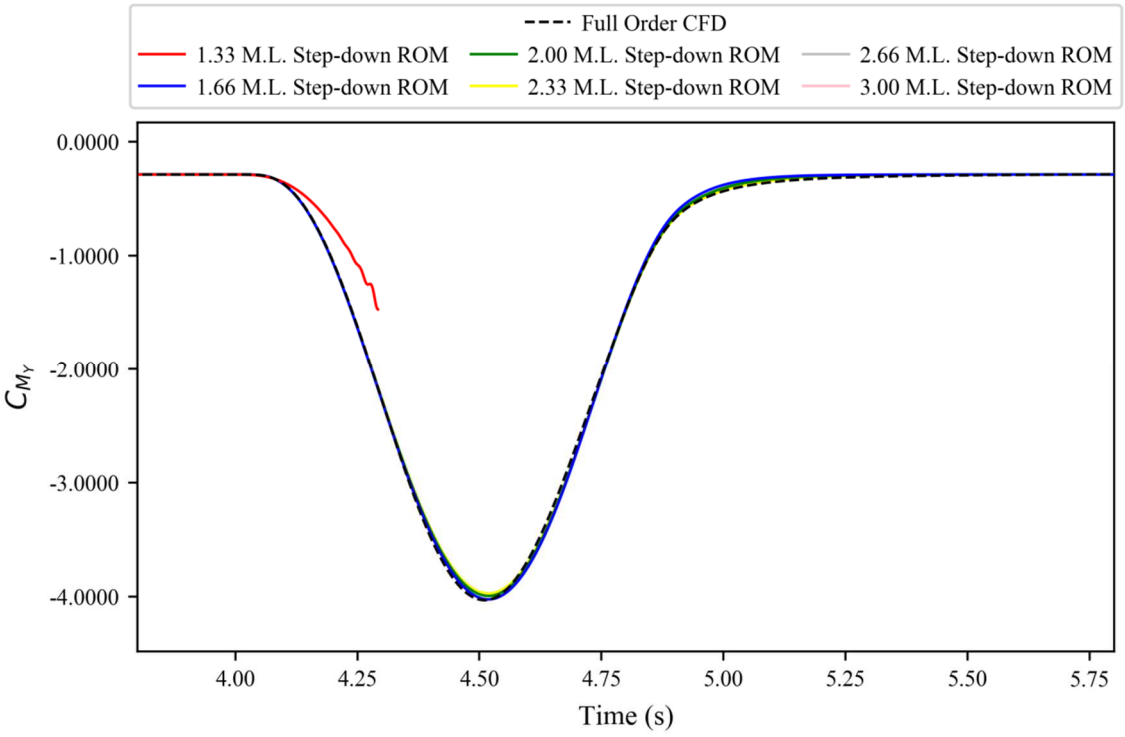

Figure 4.66 Step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

Looking at the peak response in more detail, a decision can be made on which ROM to choose as the basis going forward by verifying how the different ROMs maintain accuracy at arguably the most important part of the response.

Figures 4.67-4.74 shows that, whilst 2.00 model lengths of data produces very good results, ROM convergence had not quite been achieved for this case, and instead first occurs in the 2.33 model length case; after which further inclusion of data does not make any notable difference to the results. Conversely, the 1.33 and 1.66 model length cases produce visibly different results for all cases. For the 1.33 model length case this is due to model instability, for the 1.66 model length case the same counter intuitive behaviour seen in previous ROMs is observed; where the result often matches the CFD more accurately than longer model length cases. However, it was also shown previously to have settling inaccuracy so this behaviour is likely anomalous and specific to this case rather than down to ROM accuracy.

The results at the peak (Figures 4.67-4.74) lead to a similar conclusion to those reached for the whole response (Figures 4.59-4.66). Running the sharp-edged gust until the gust reaches 2.33 model lengths post-impact appears to be sufficient to maintain the accuracy of the ROM whilst minimising the computational cost of building the ROM initially.

It is worth noting that on the peak response figures (Figures 4.67-4.74), some of the responses may not be visible if the peak accuracy was extremely poor; additionally extremely similar responses can be hard to distinguish from one another. Highly unstable responses have also been removed from the peak response figures as their oscillations make the plots difficult to read.

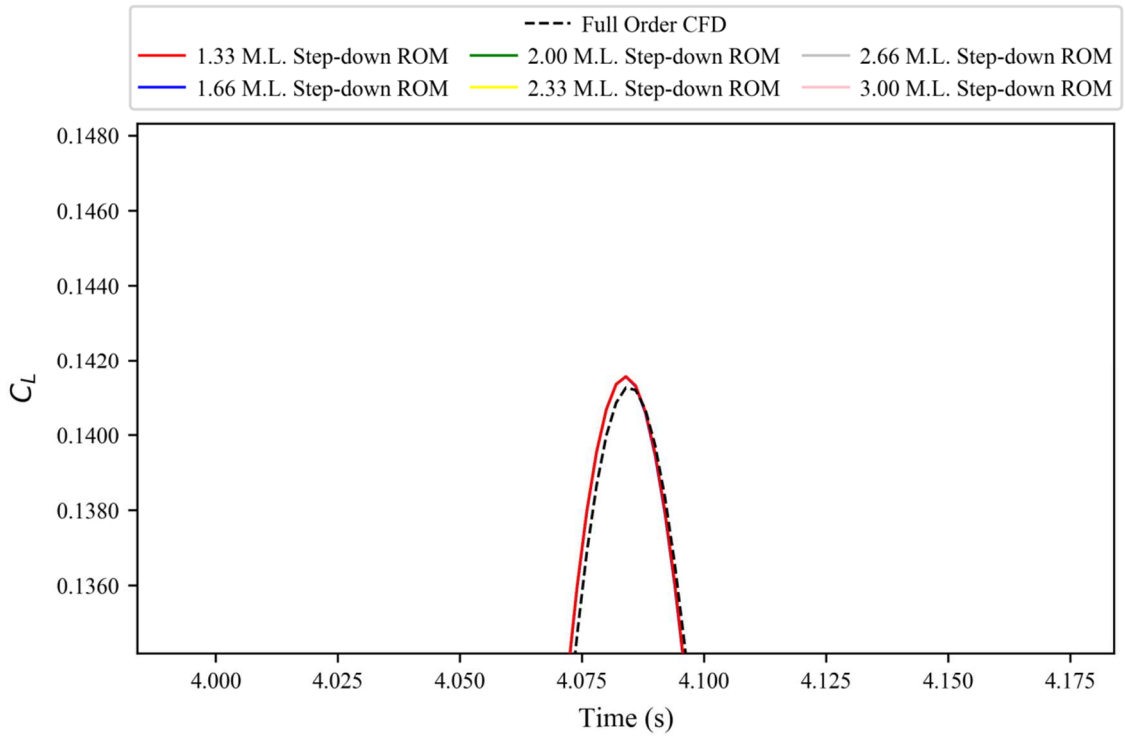

Figure 4.67. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 1.

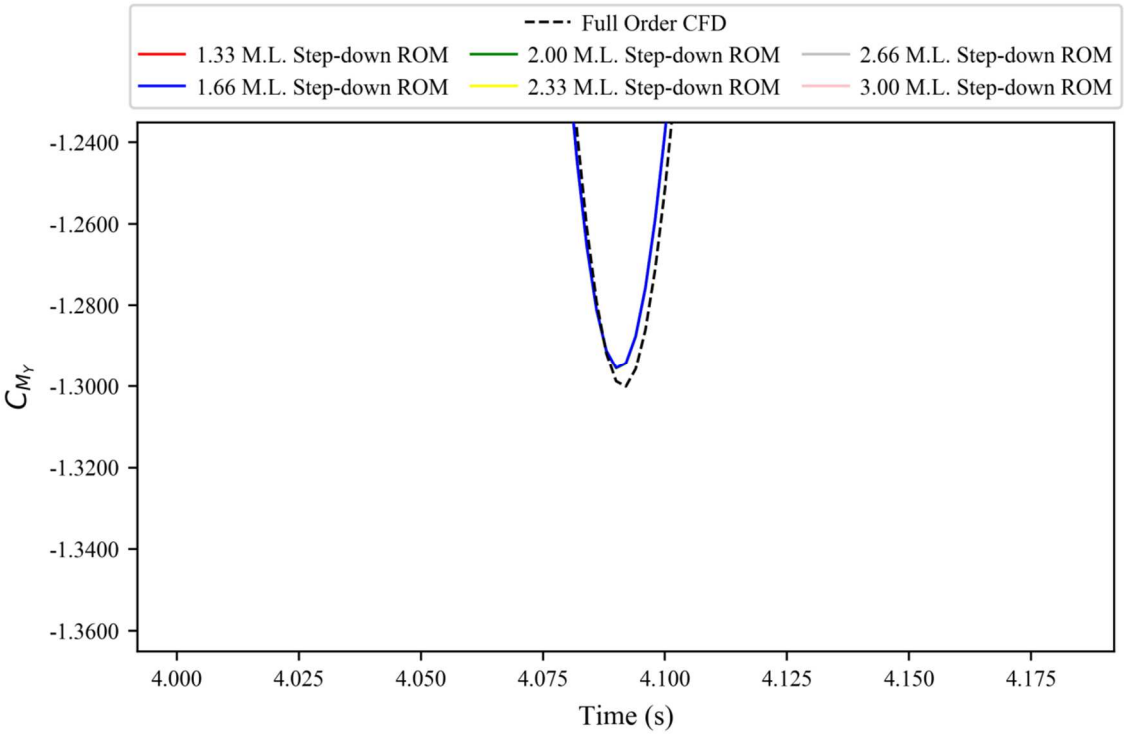

Figure 4.68. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 1.

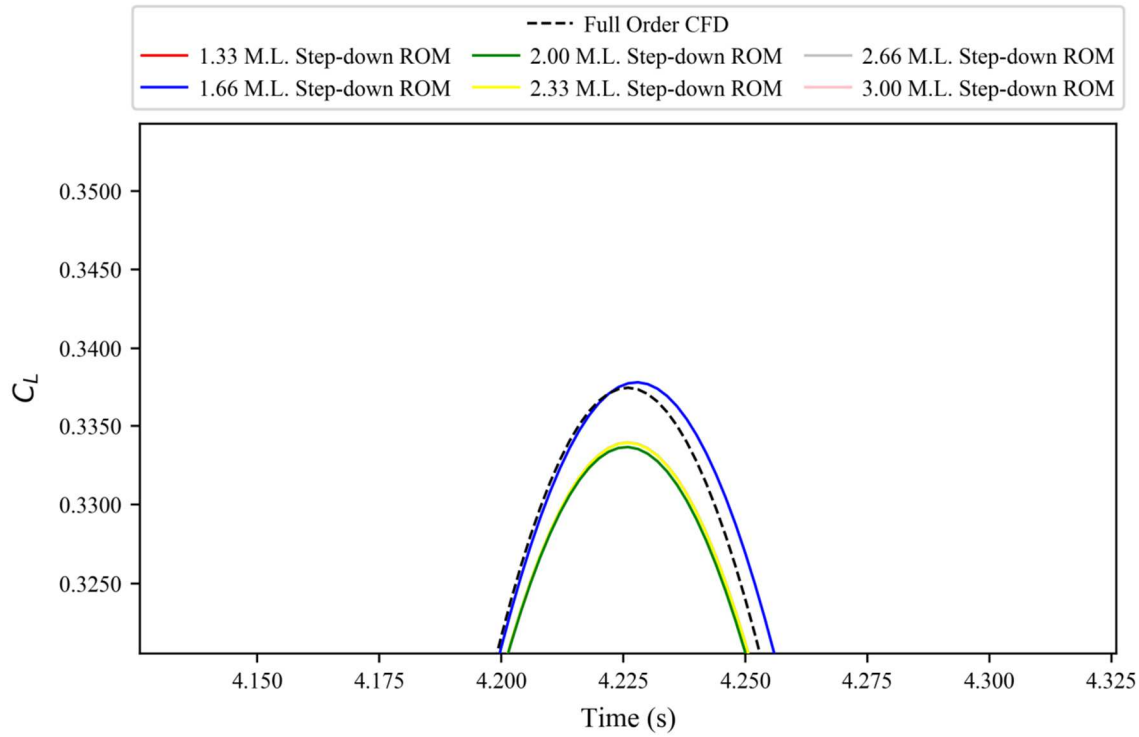

Figure 4.69. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 2.

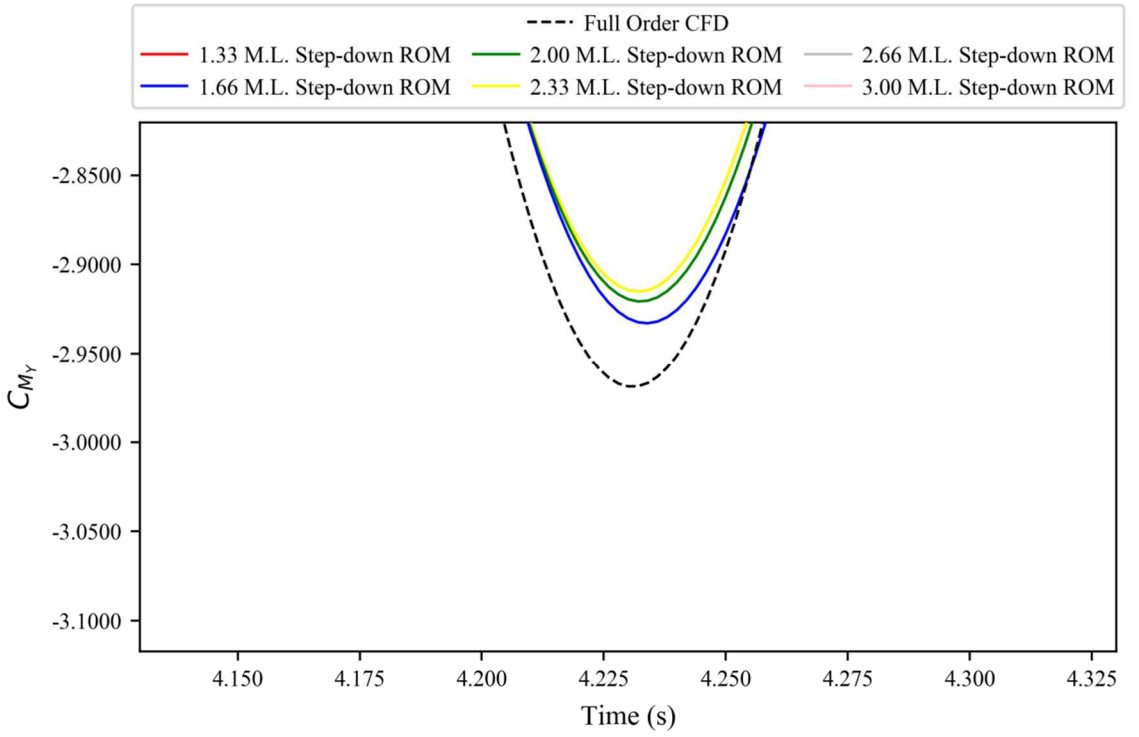

Figure 4.70. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 2.

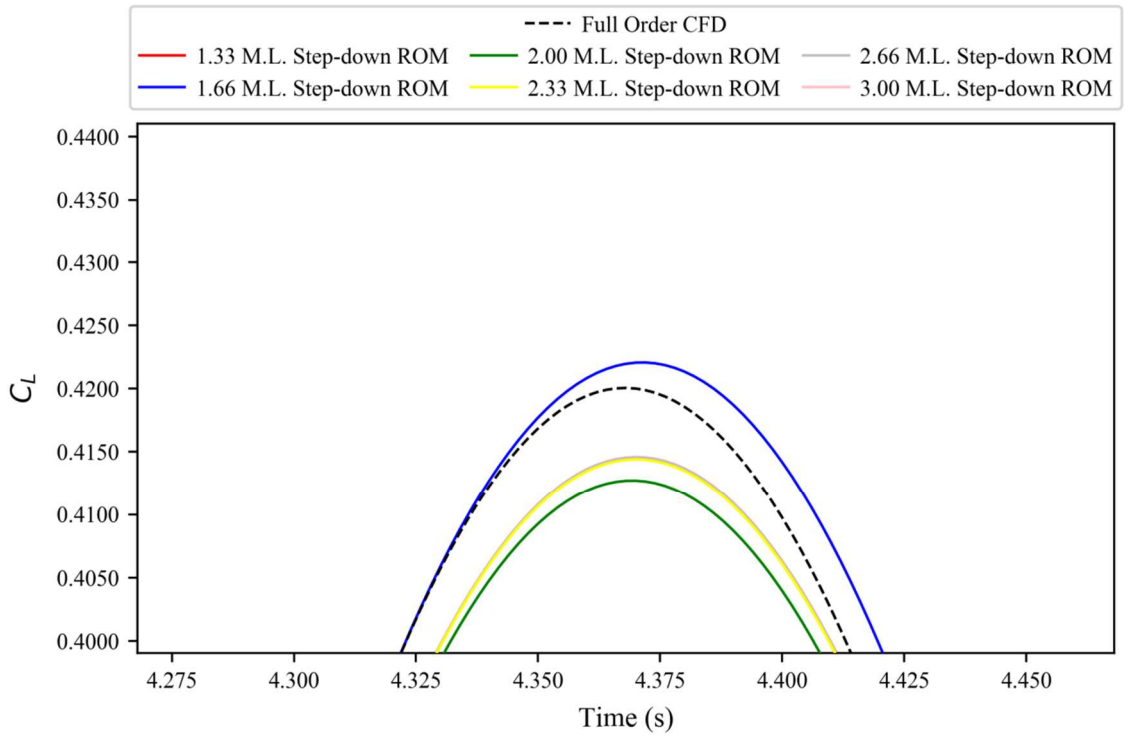

Figure 4.71. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 3.

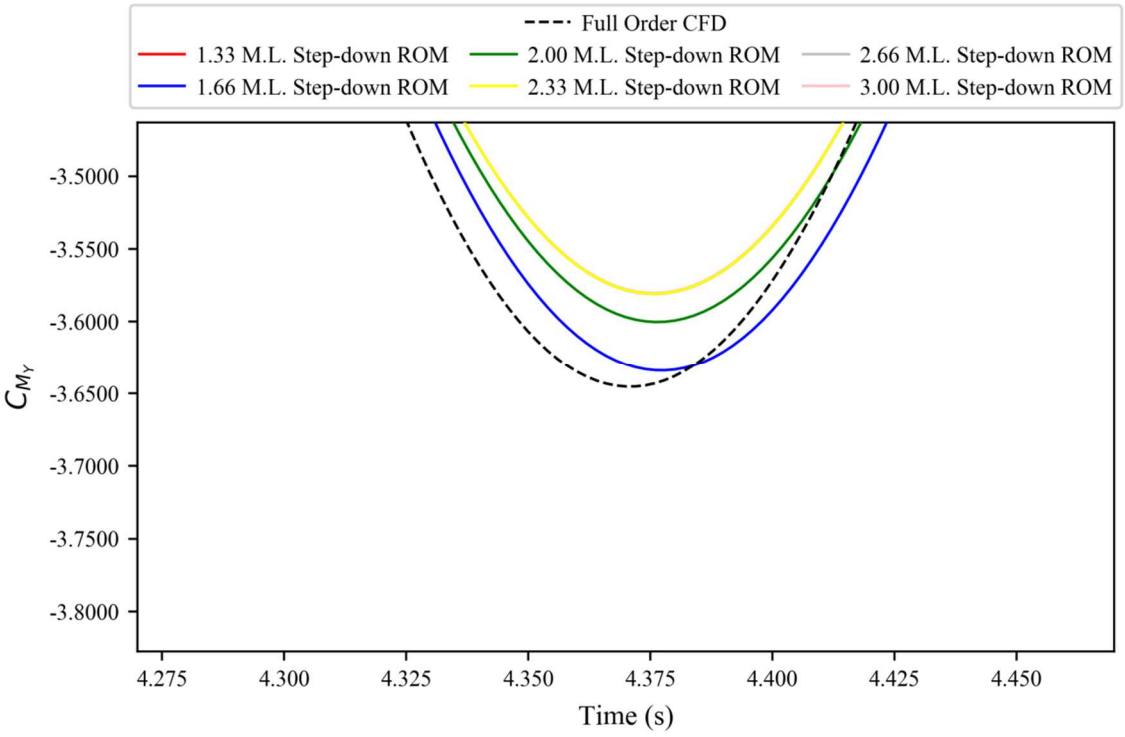

Figure 4.72. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 3.

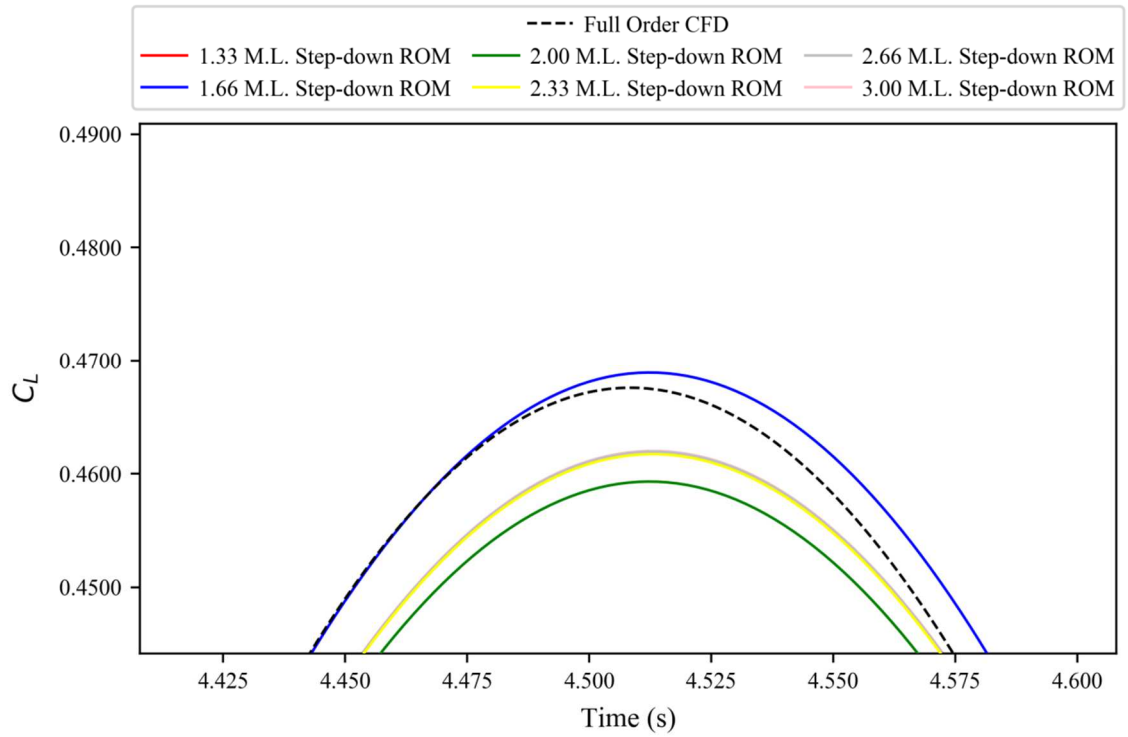

Figure 4.73. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 2, gust case 4.

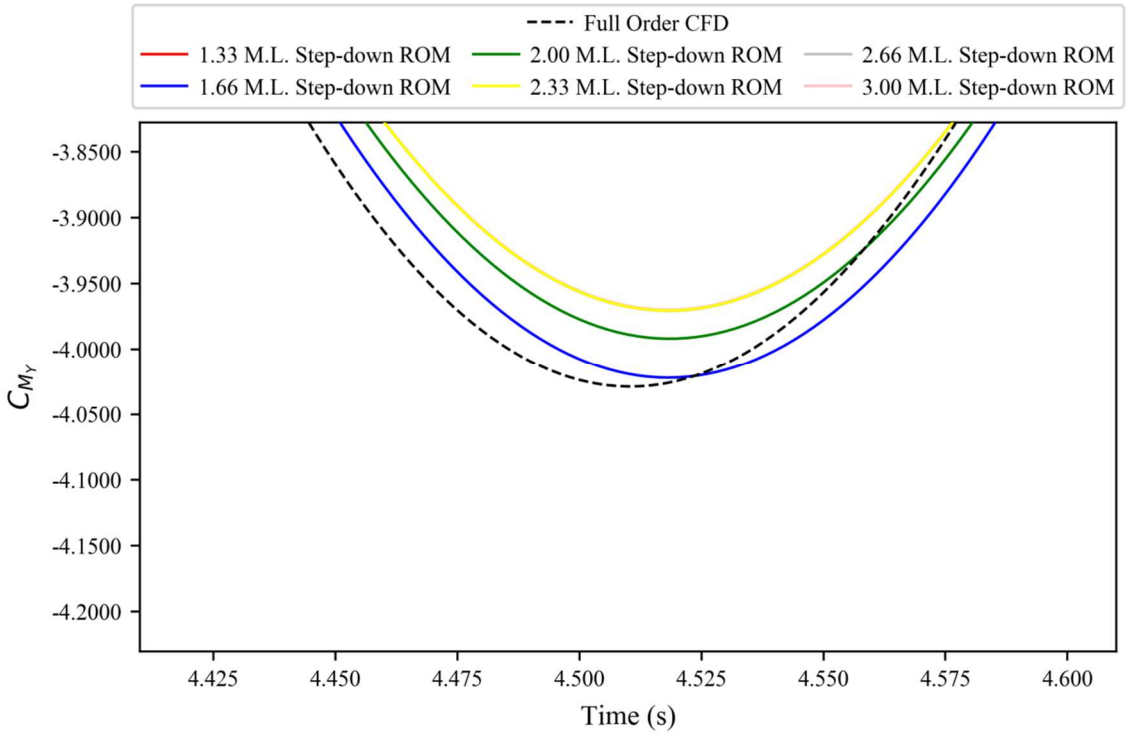

Figure 4.74. Peak response for the step-down based ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 2, gust case 4.

It is hard to fully quantify the computational savings made, as these will vary from case to case. However, it is possible to give some idea of the savings possible by looking at the number of time steps used after the sharp-edged gust impacts the model for this

particular case. If considering only the time steps after the sharp-edge gust impacts the model, the previous ROMs (base, single sharp-edged and variable time step) used 450 time steps. The chosen 2.33 model length ROM however only uses 87 time steps. Whilst not entirely correct (due to the number of sub-iterations that are required to converge for each time step), it is reasonable to assume each time step requires the same computational cost. Under this assumption, this represents an 80%+ reduction in the computational cost in running the sharp-edged gust once it has impacted the model.

4.5. Restarting

4.5.1. Motivation and Implementation

As discussed in Section 1.5.5, stability is an important aspect of any ROM. For discrete ROMs, such as the ones put forward in this thesis, stability is defined as all eigenvalues of the system having an absolute value less than one. However, whilst unstable eigenvalues may exist, also worth noting is that this is not always obvious from the output response. Typically ROM instability will cause the system response to quickly experience run-away growth; often oscillatory in nature and commonly exponential in size. Yet, for some cases the system output response may be reasonable, despite system instability; although this is not common.

Being able to ensure ROM stability has a large impact on the ROM robustness. It is extremely difficult, if not impossible, to consistently predict which setups will lead to the construction of an unstable ROM; with the number of Markov parameters used, the flight point, etc. all potentially having an impact. Therefore, being able to ensure the ROM being created is stable ensures that a useable ROM can be created without having to potentially rerun the sharp-edge gust to obtain extra Markov parameters, or other similar requirements.

To implement restarting, the discrete method as laid out in Section 3.4.1 is coded into the script that executes the initial construction of the ROM. Once the three system matrices ($\widetilde{\mathbf{A}}_{red}$, $\widetilde{\mathbf{B}}_{red}$, $\widetilde{\mathbf{C}}_{red}$) are created via ERA, the eigenvalues of the $\widetilde{\mathbf{A}}_{red}$ matrix are obtained and ordered, by absolute value, in descending order. The eigenvalue with the greatest absolute value is regarded as the worst eigenvalue and is used to determine if restarting is required. If it is (due to this worst eigenvalue having an absolute value

equal to or greater than one), then the restarting method is applied to construct a new Hankel matrix which is then used to obtain a new set of system matrices. The eigenvalues of the new $\widetilde{\mathbf{A}}_{red}$ matrix are then calculated and if they are all within the stable region, then restarting is complete and the new system matrices can be used to calculate the output system response. If the new eigenvalues contain unstable values, then the processes is repeated until all eigenvalues are stable.

4.5.2. Results

As seen earlier (Figures 4.59-4.66) the step-down ROM created from the sharp-edged gust that only ran for 1.33 model lengths post-impact, was unstable. However, this only utilised 49 Markov parameters, and given that at least 41 are needed to construct a size-20 ROM, this does not leave many parameters available for use in restarting. Therefore a new ROM was chosen to test this restarting method; one that involves running the sharp-edged gust for 1.40 model lengths post-impact.

This ROM has been set up as a SISOSISO ROM, and therefore to obtain the system response for both the coefficient of lift and the coefficient of pitching moment, two sets of system matrices must be constructed; and thus two separate sets of eigenvalues have to be assessed and, if necessary, restarted.

Figures 4.75 and 4.77 show the starting eigenvalues for the coefficient of lift and the coefficient of pitching moment ROMs (respectively). In both cases, the majority of the eigenvalues fell within the stable region (visualised by the circle with a radius of 1 and centred on the origin), with only a handful being unstable. Both cases only required a single restart in order to obtain a complete set of stable eigenvalues (Figures 4.76 and 4.78).

If these four ROMs (original/unstable and new/stable for both force coefficients) are then used to obtain the system response to the four gust cases for flight point 2, the impact restarting has had can be assessed.

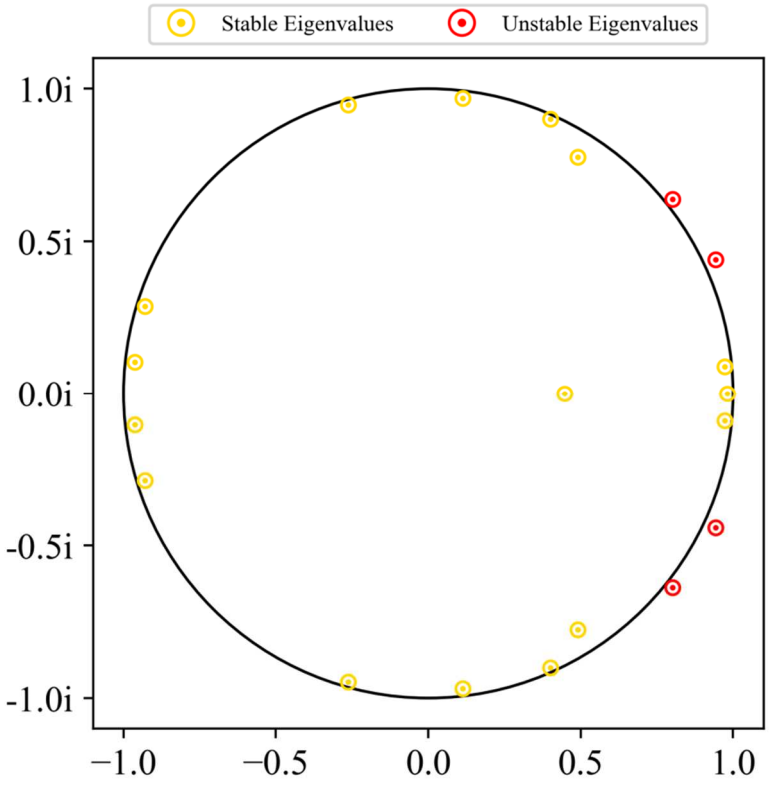

Figure 4.75. Starting eigenvalues for a discretely unstable ROM constructed to model the system's coefficient of lift.

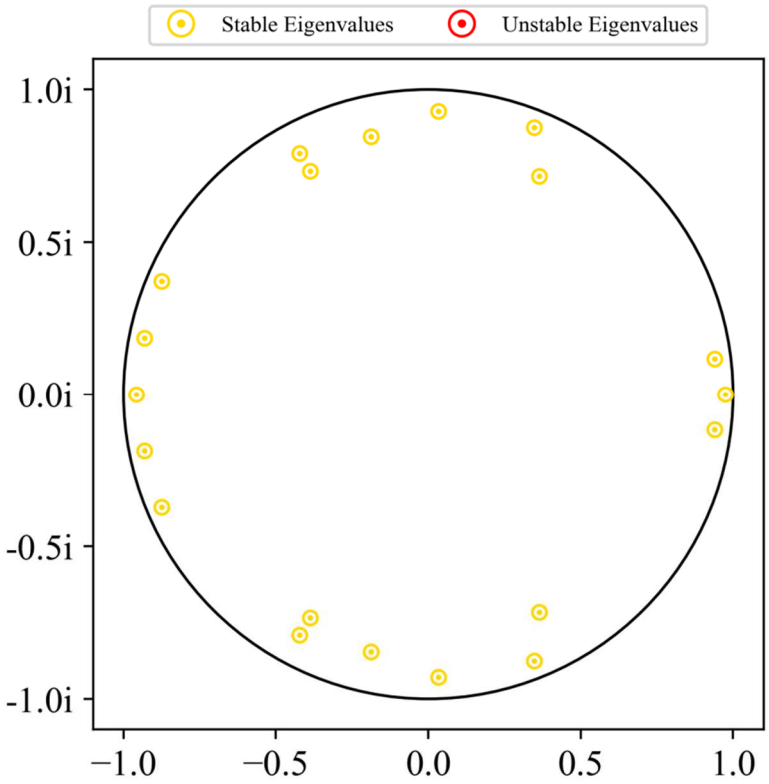

Figure 4.76. Final eigenvalues, after restarting for a now discretely stable ROM constructed to model the system's coefficient of lift.

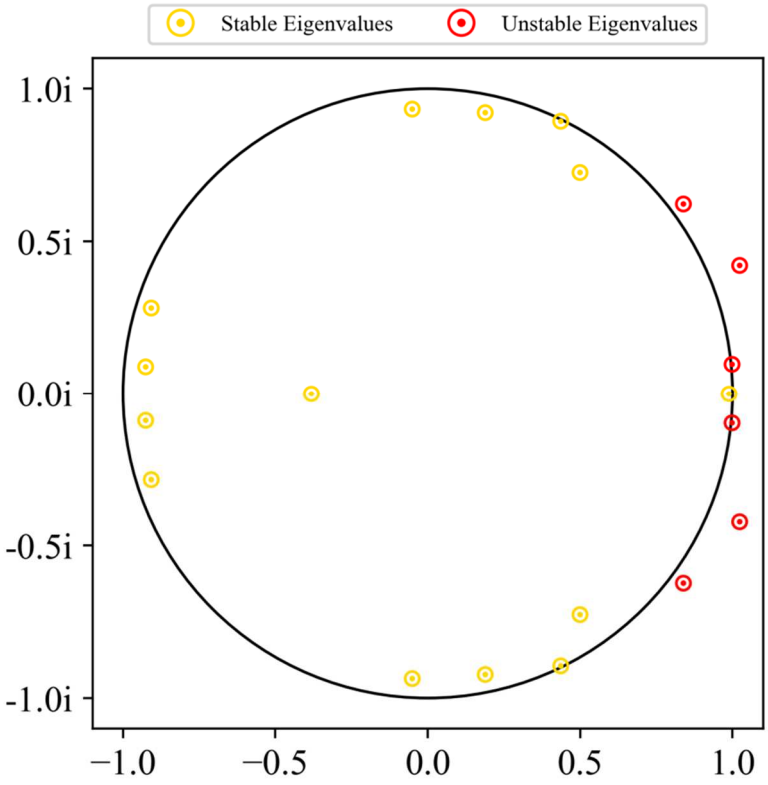

Figure 4.77. Starting eigenvalues for a discretely unstable ROM constructed to model the system's coefficient of pitching moment.

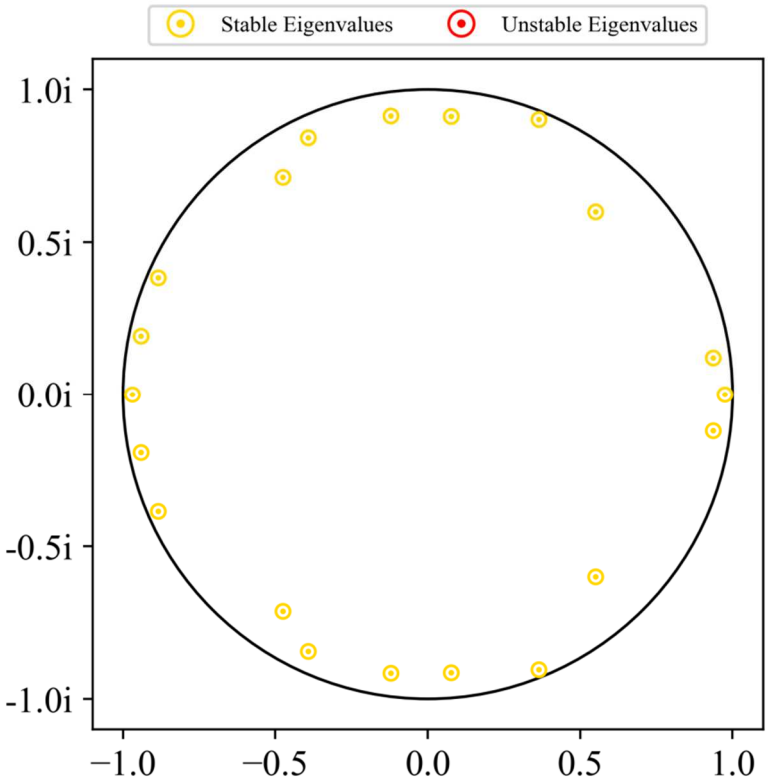

Figure 4.78. Final eigenvalues, after restarting for a now discretely stable ROM constructed to model the system's coefficient of pitching moment.

Figures 4.79-4.86 clearly demonstrate the positive impact that restarting has had. Before restarting, the ROMs are highly unstable and demonstrate an oscillating, exponential

growth that renders the ROMs entirely useless. However, for the post-restarting ROMs, this behaviour is no longer present and the results are a reasonable match to the full order CFD; certainly as good as could be expected given the ROM was built using 0.93 model lengths worth of time steps fewer than was previously identified ideal (see Section 4.4.5).

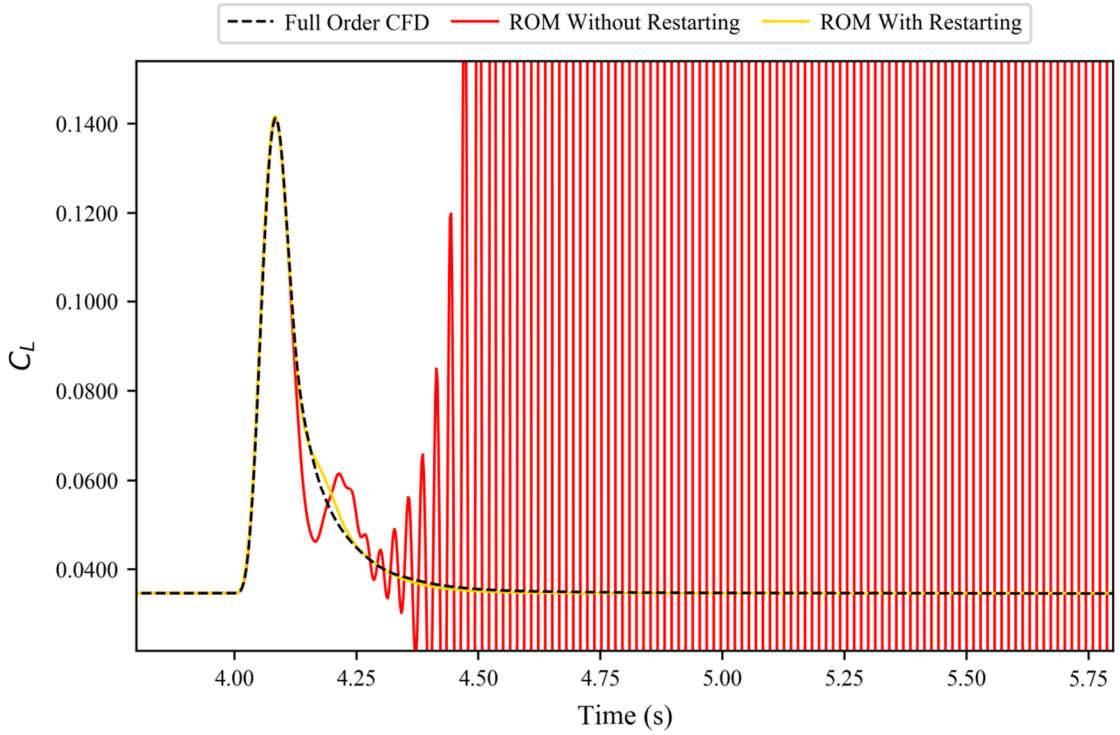

Figure 4.79. Coefficient of lift response for the FFAST wing at flight point 2, gust case 1. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

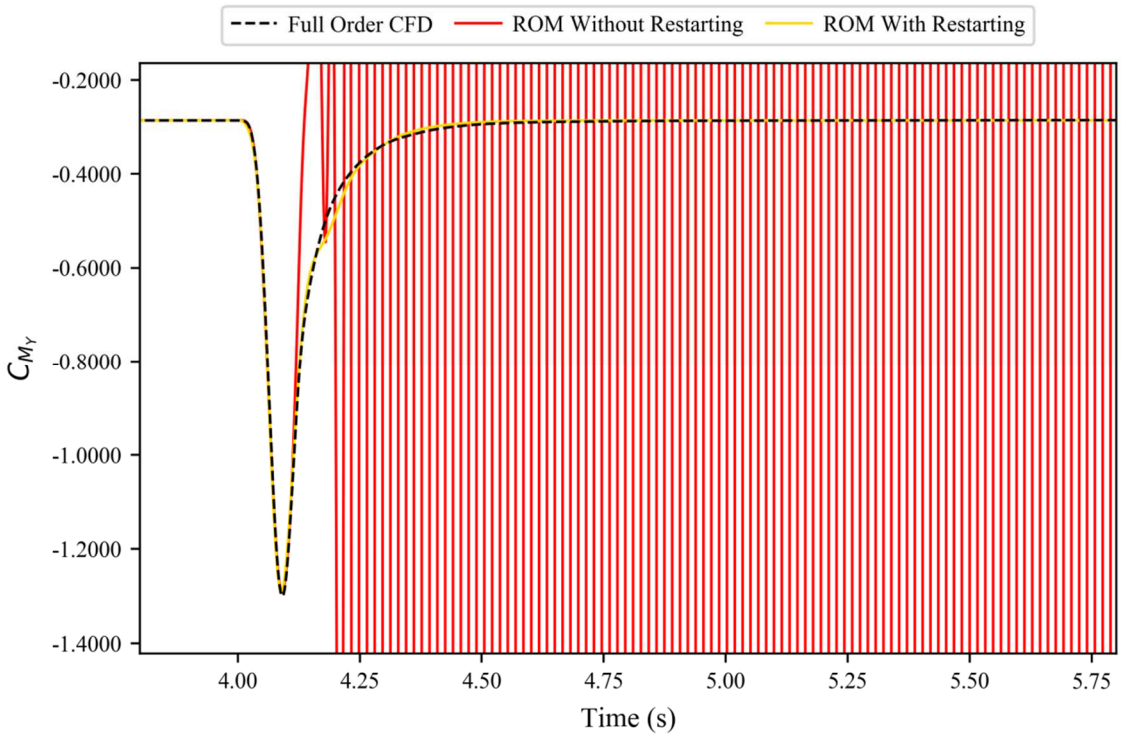

Figure 4.80. Coefficient of pitching moment response for the FFAST wing at flight point 2, gust case 1. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

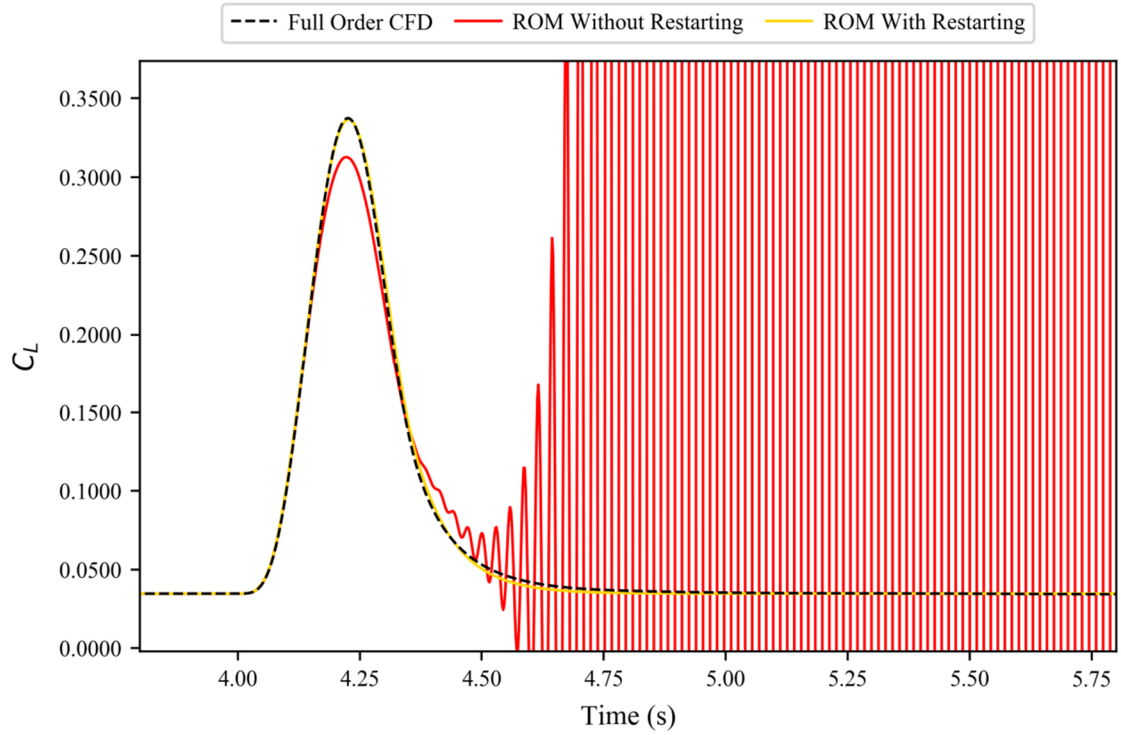

Figure 4.81. Coefficient of lift response for the FFAST wing at flight point 2, gust case 2. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

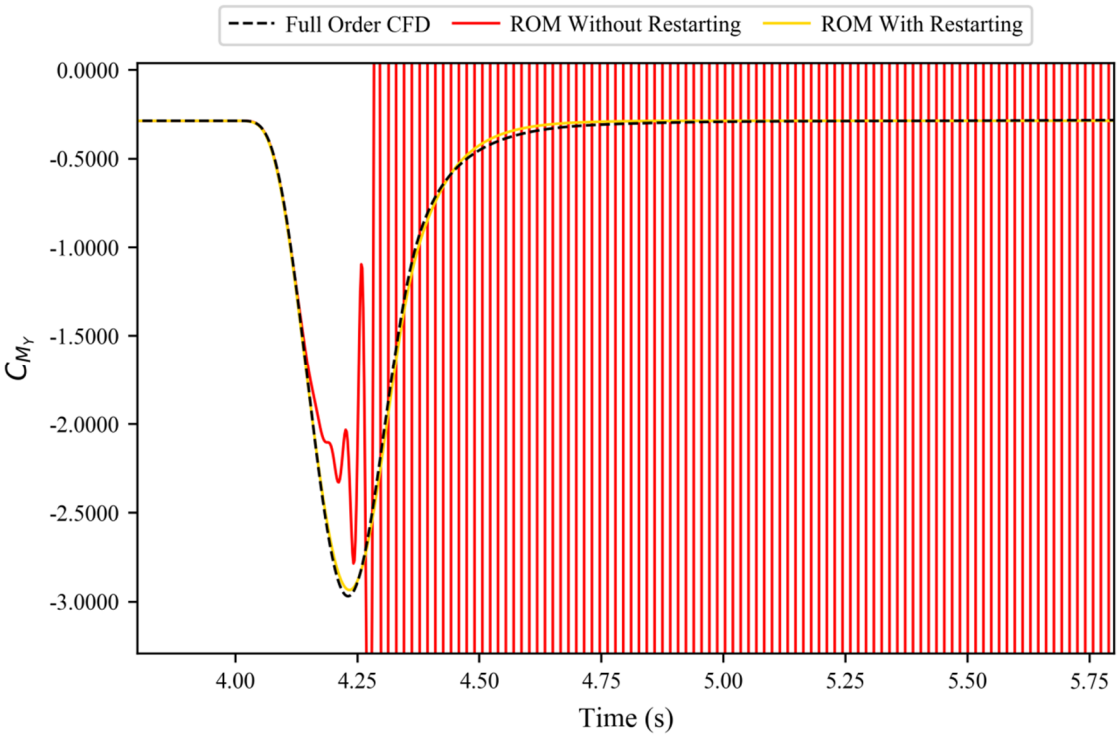

Figure 4.82. Coefficient of pitching moment response for the FFAST wing at flight point 2, gust case 2. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

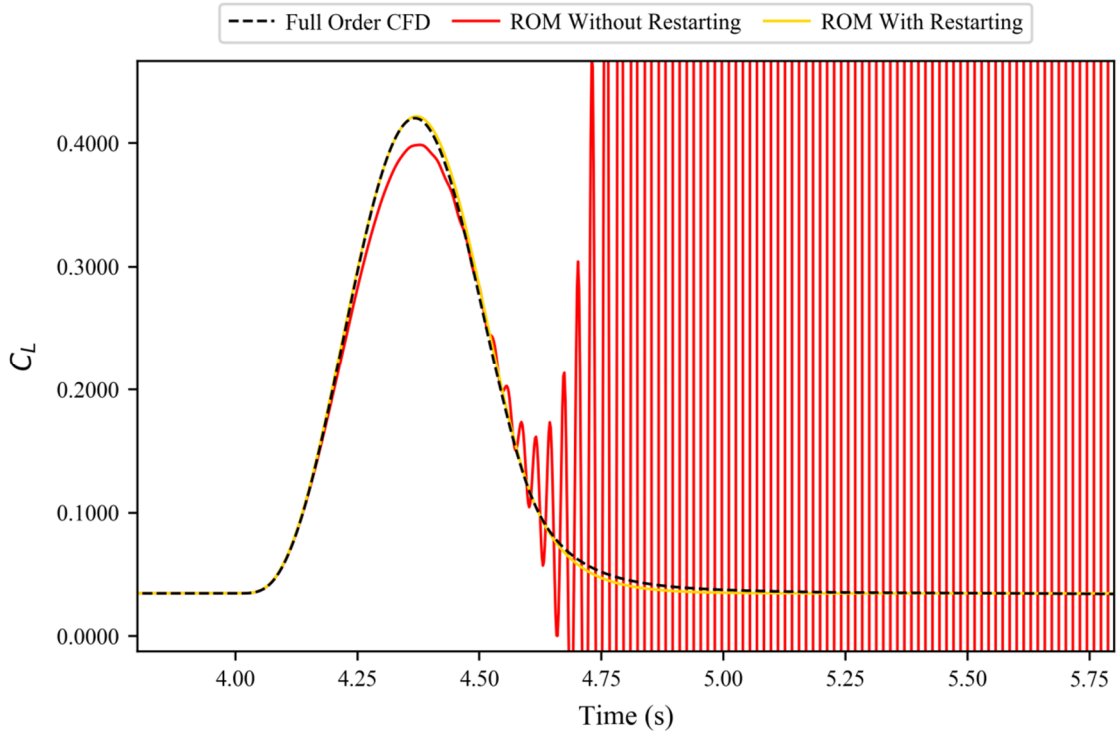

Figure 4.83. Coefficient of lift response for the FFAST wing at flight point 2, gust case 3. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

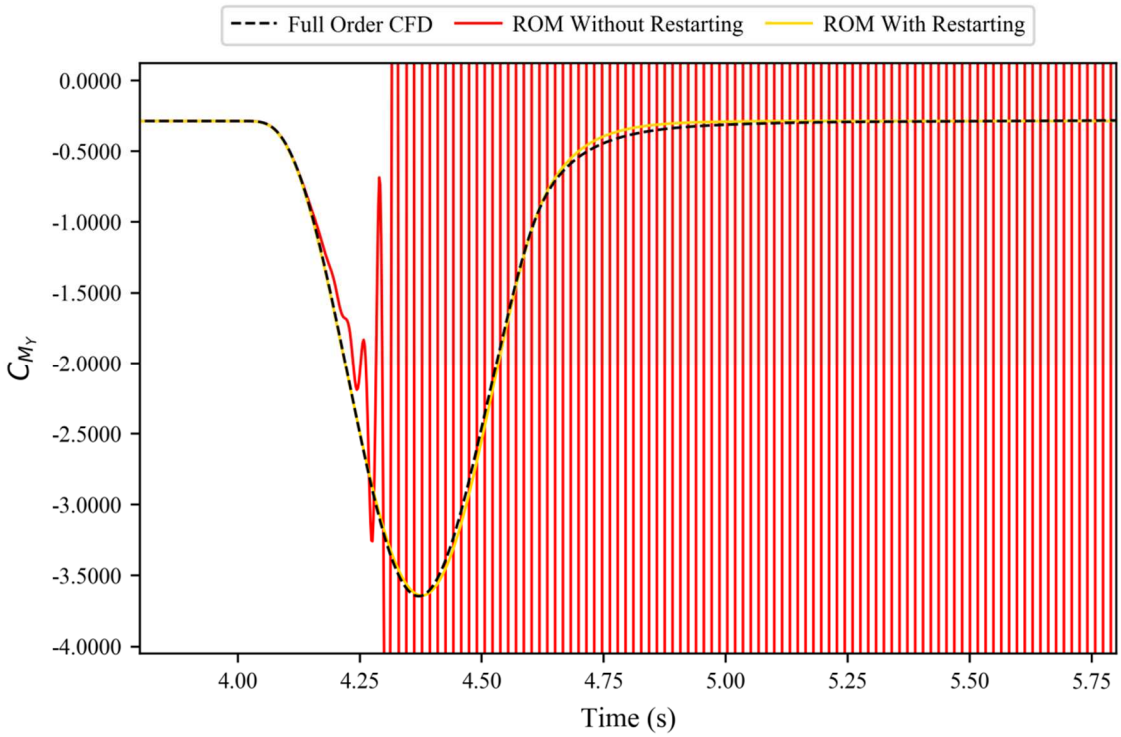

Figure 4.84. Coefficient of pitching moment response for the FFAST wing at flight point 2, gust case 3. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

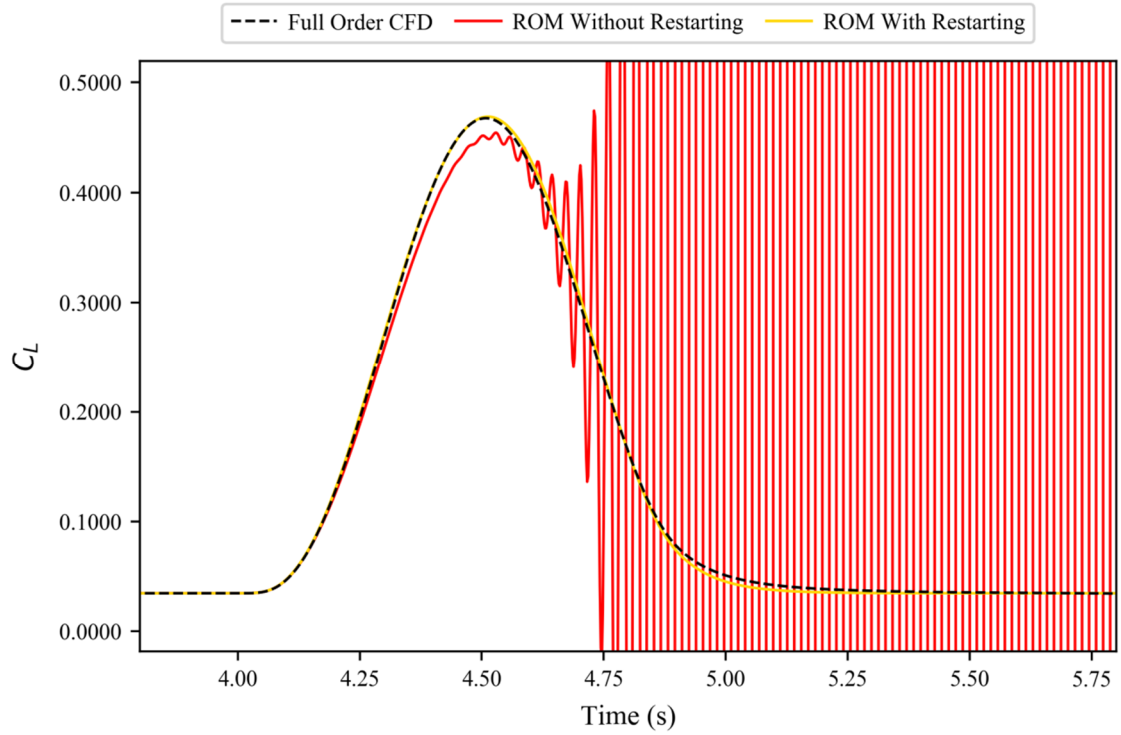

Figure 4.85. Coefficient of lift response for the FFAST wing at flight point 2, gust case 4. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

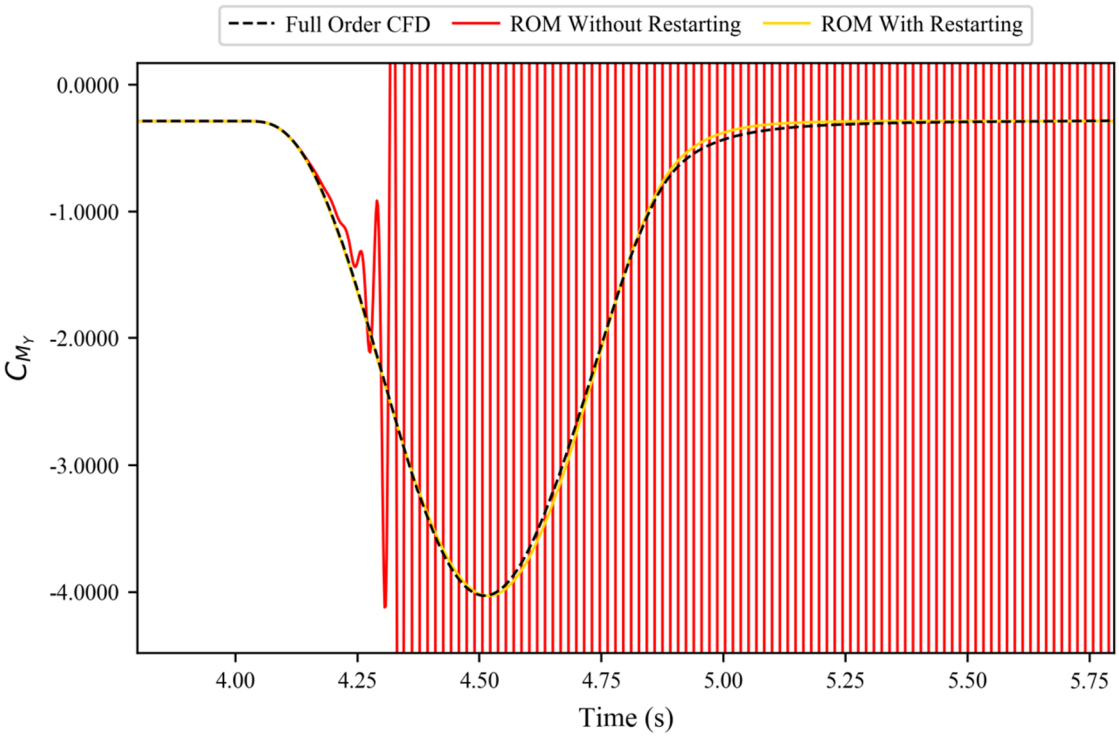

Figure 4.86. Coefficient of pitching moment response for the FFAST wing at flight point 2, gust case 4. Compared are the results from full order CFD, a discretely unstable ROM and the same ROM made stable via restarting.

4.6. Additional Results

The final ROMs presented in this chapter of the thesis are step-down based ROMs built using a single sharp-edged gust of magnitude $1ms^{-1}$, which transitions from large time steps to small ones 0.2 MAC lengths ahead of the leading edge, continues to run for 2.33 model lengths post-impact and with a ROM-size of 20. As with all previous ROMs presented, each ROM is only valid at the flight point at which it was created. Stability is ensured via restarting for this ROM.

Whilst this ROM method has been extensively demonstrated on the four gusts of flight point 2 (see Figures 4.59-4.74) it is worth exploring the other four flight points laid out in Table 3 in order to gain a more complete understanding on how well the ROM method performs.

4.6.1. Flight Point 1

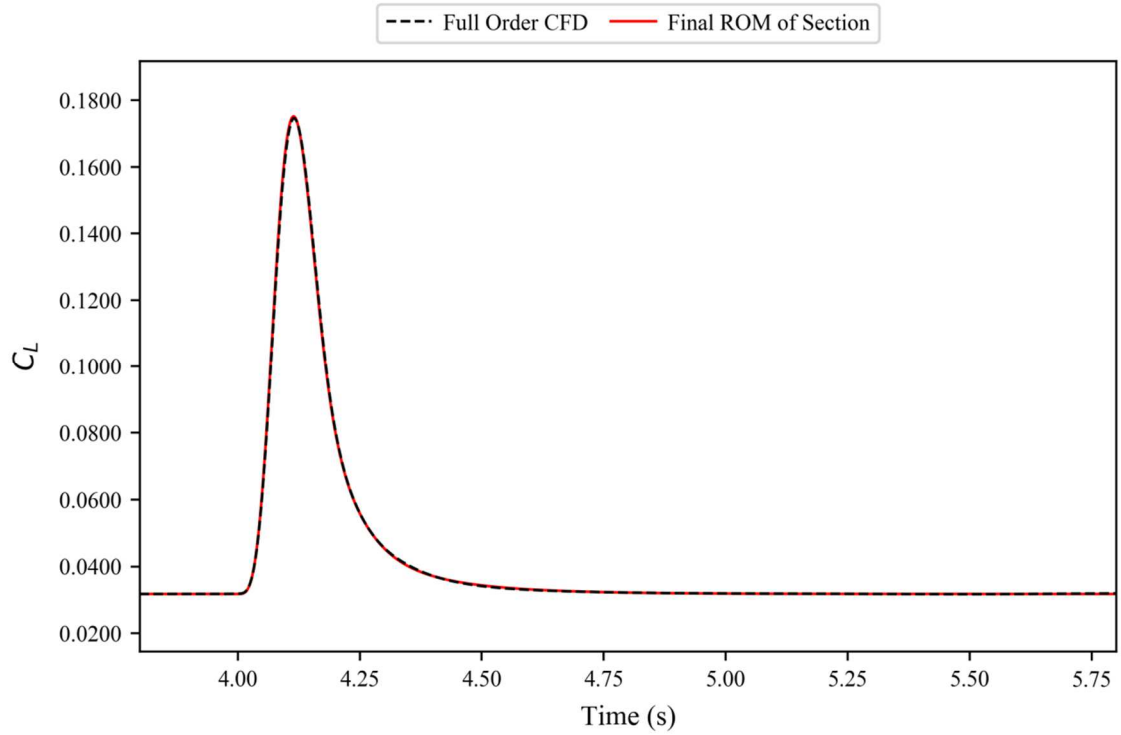

Figure 4.87. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 1, gust case 1.

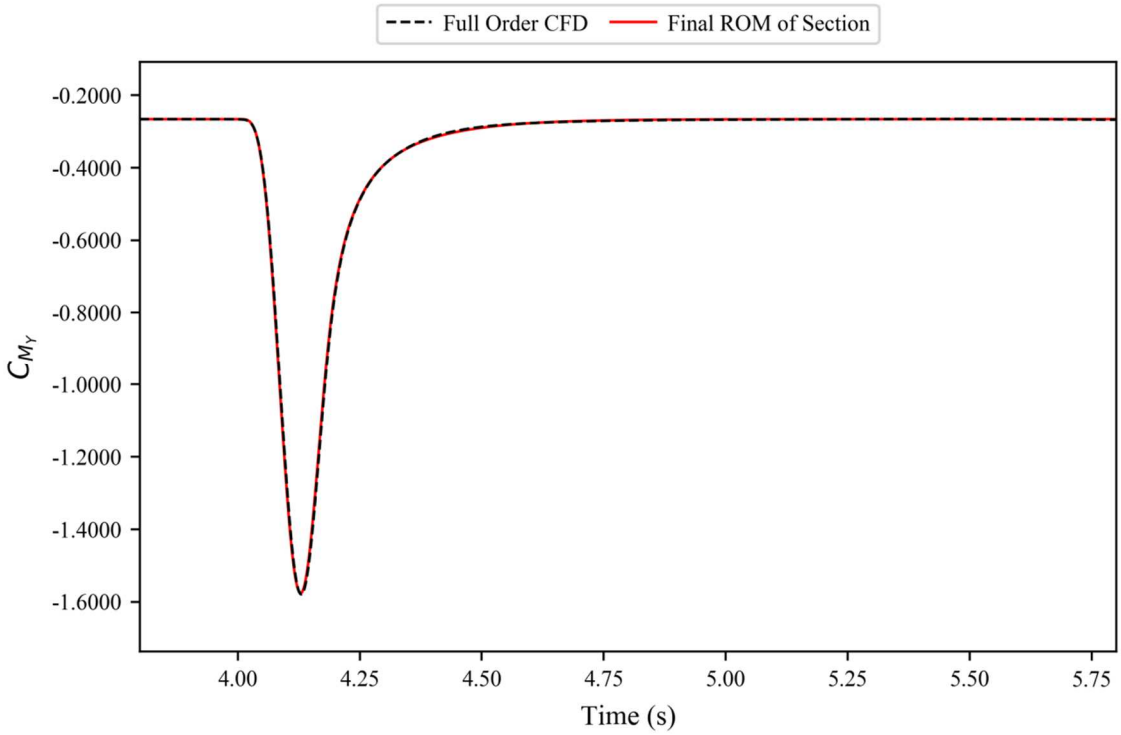

Figure 4.88. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 1, gust case 1.

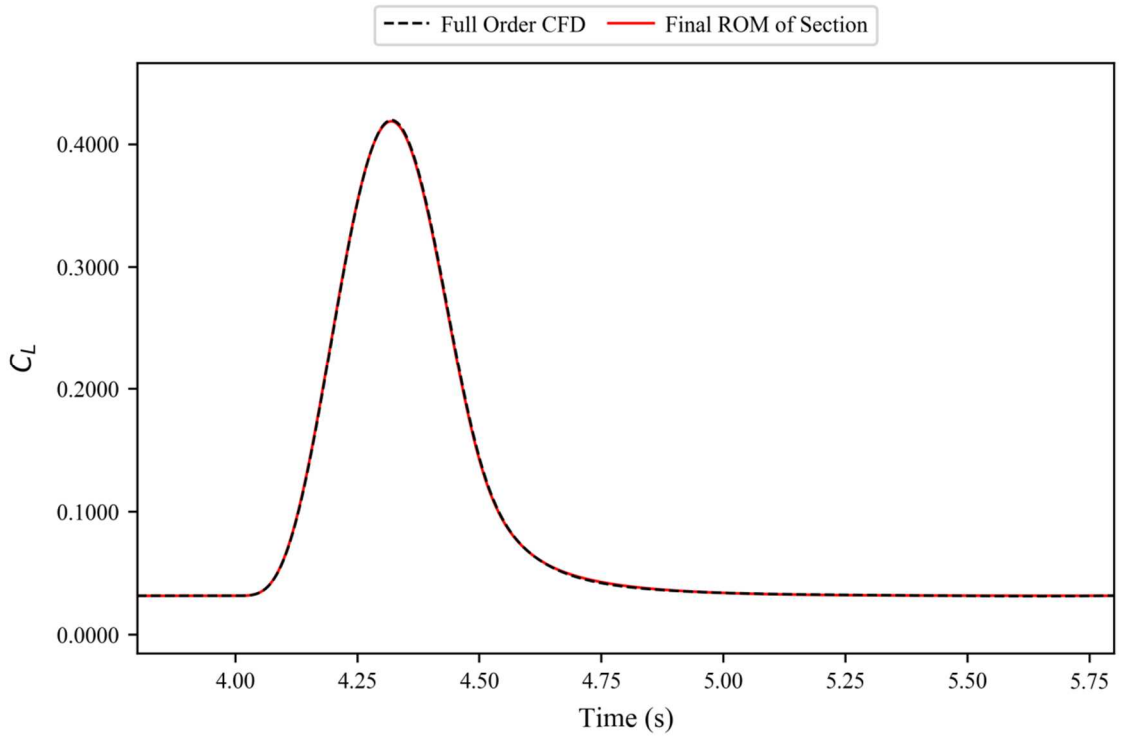

Figure 4.89. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 1, gust case 2.

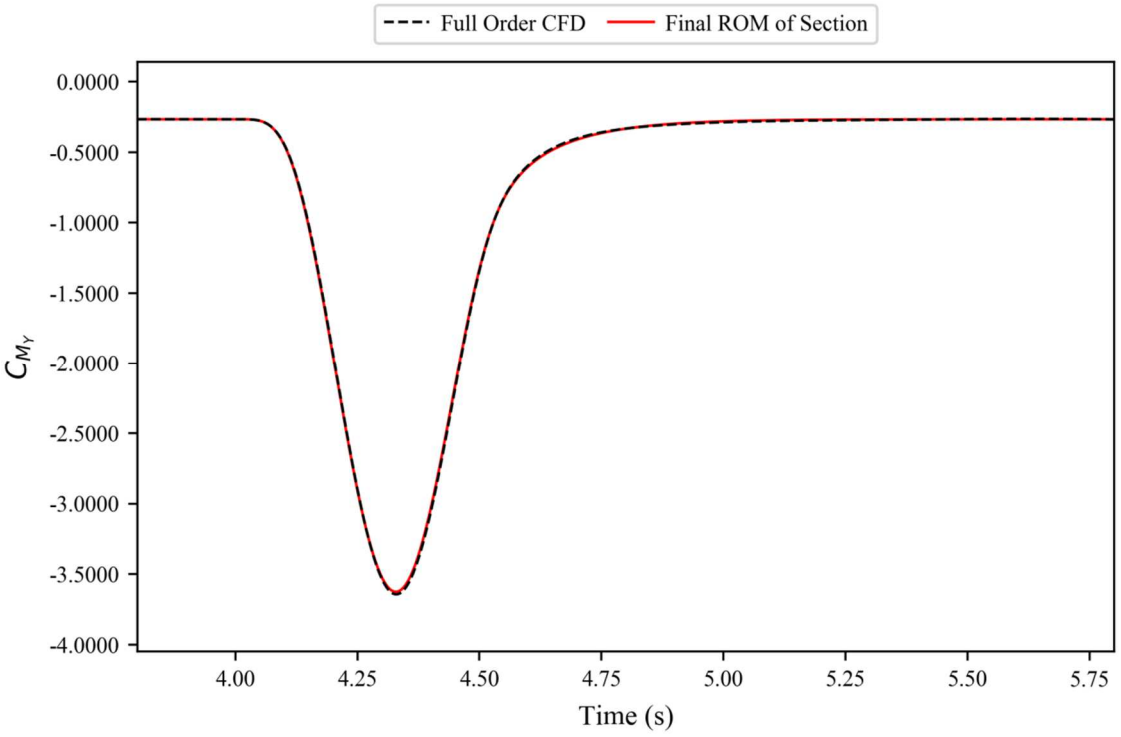

Figure 4.90. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 1, gust case 2.

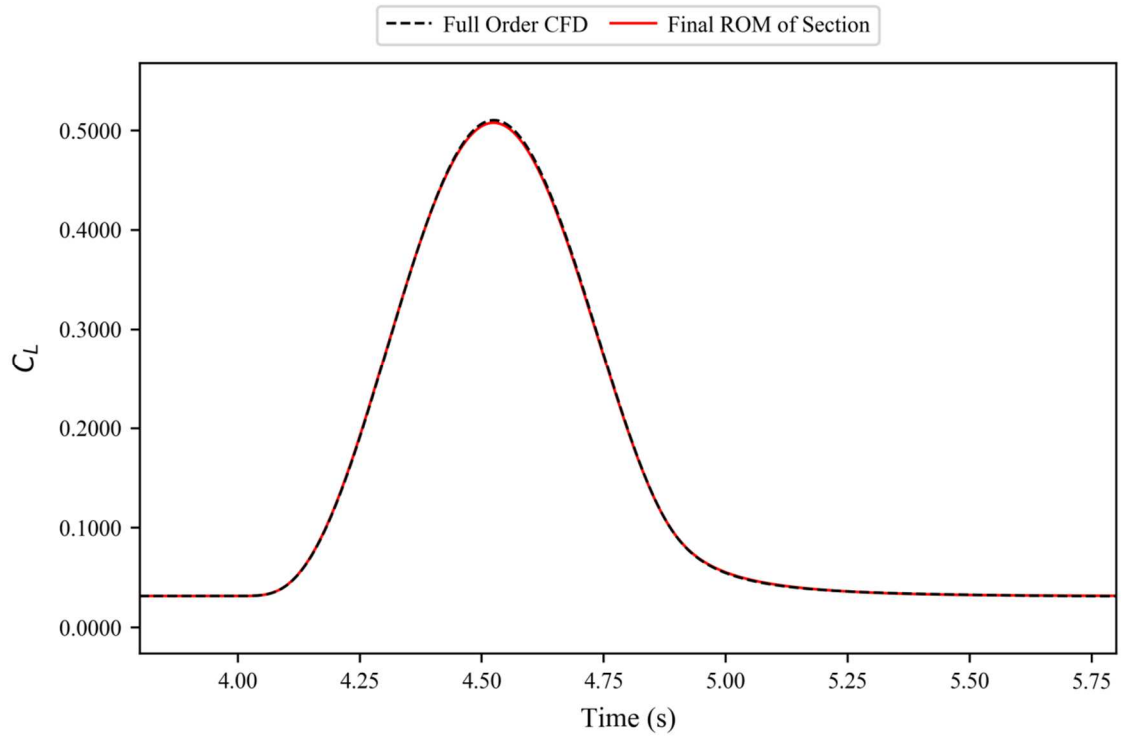

Figure 4.91. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 1, gust case 3.

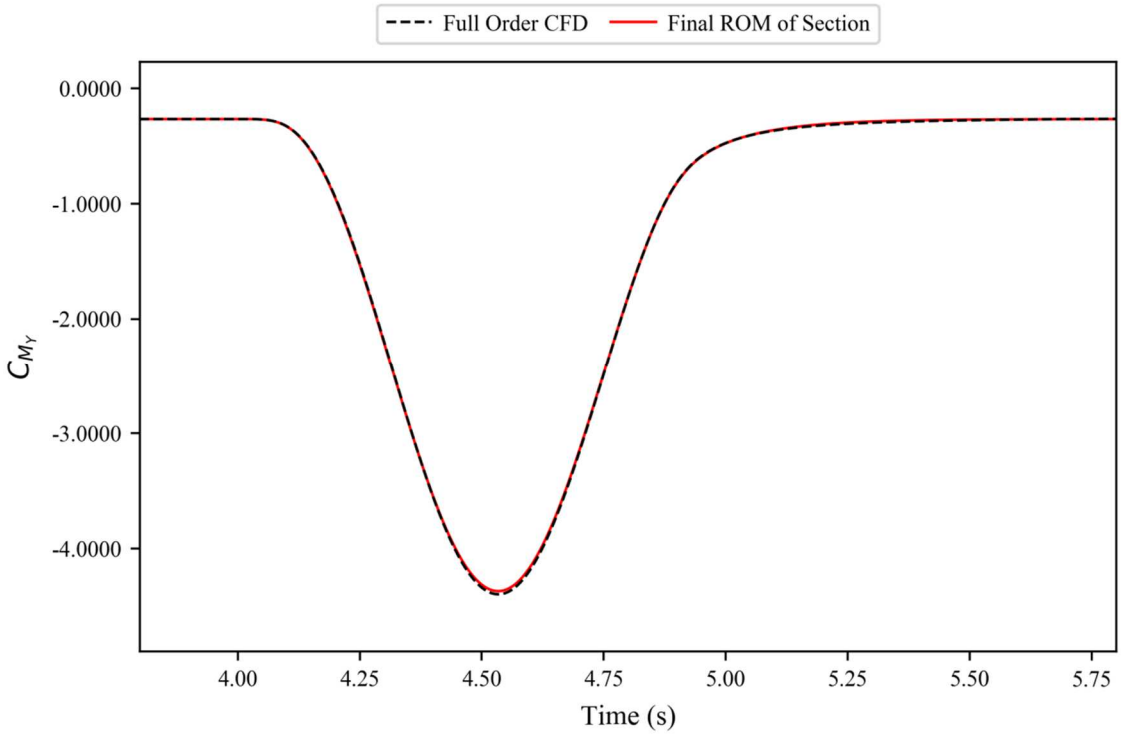

Figure 4.92. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 1, gust case 3.

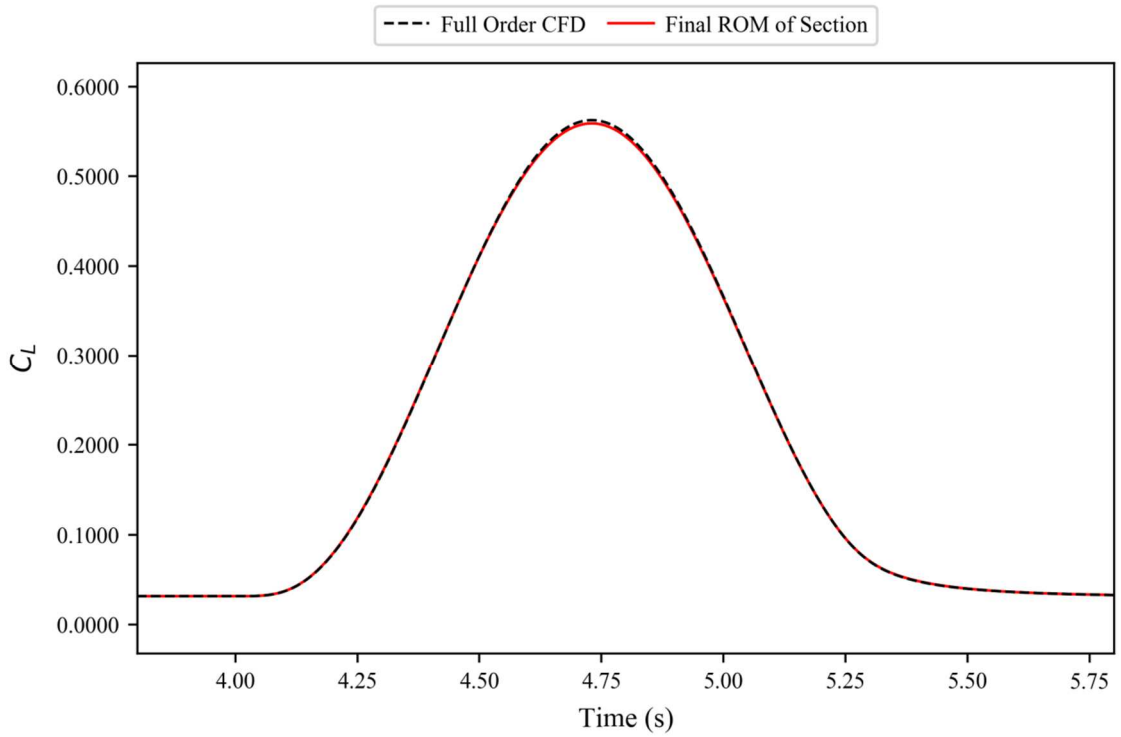

Figure 4.93. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 1, gust case 4.

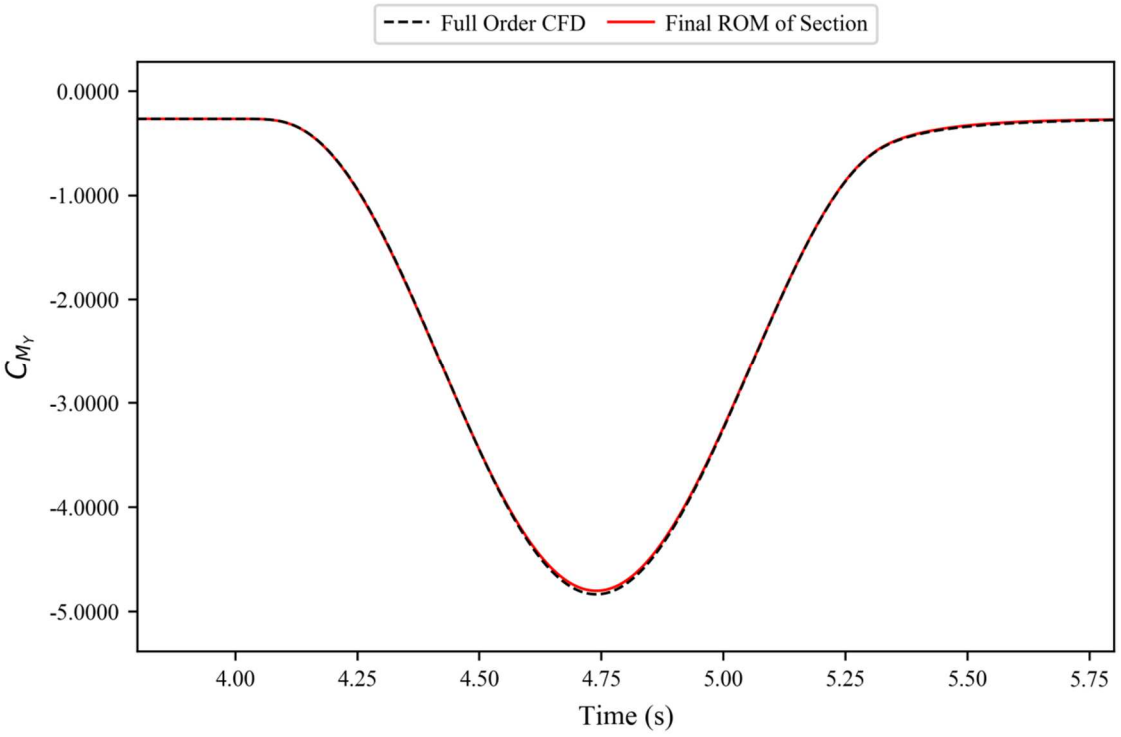

Figure 4.94. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 1, gust case 4.

4.6.2. Flight Point 3

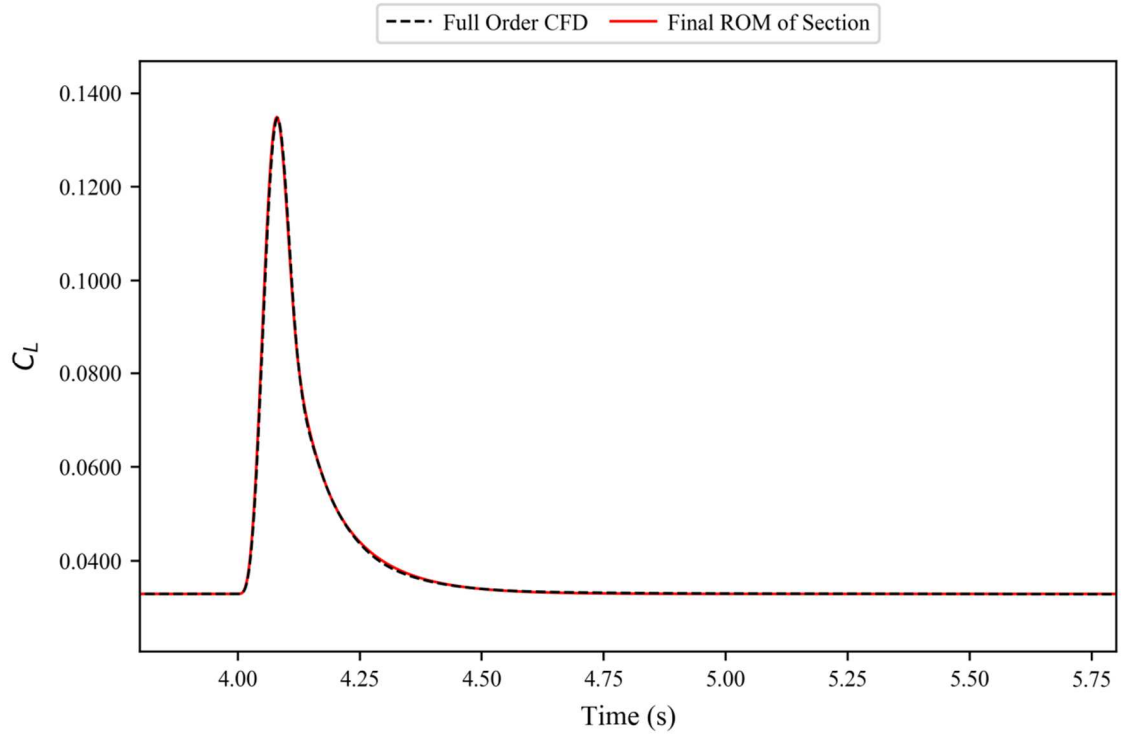

Figure 4.95. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 3, gust case 1.

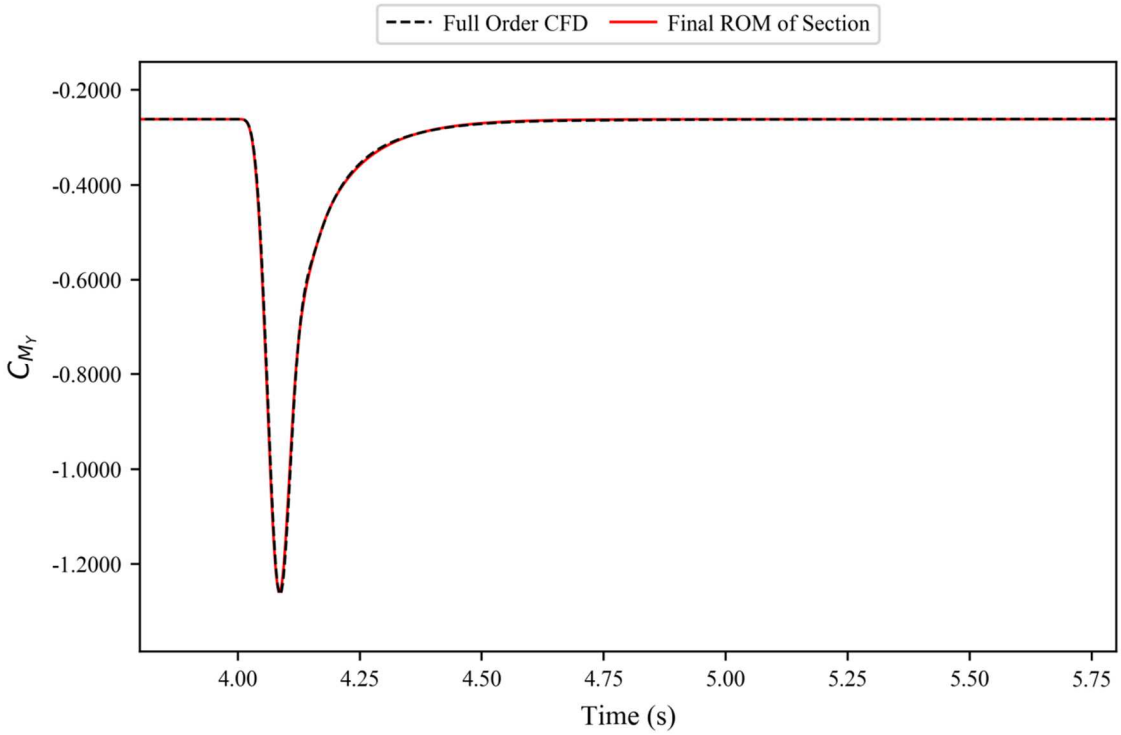

Figure 4.96. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 3, gust case 1.

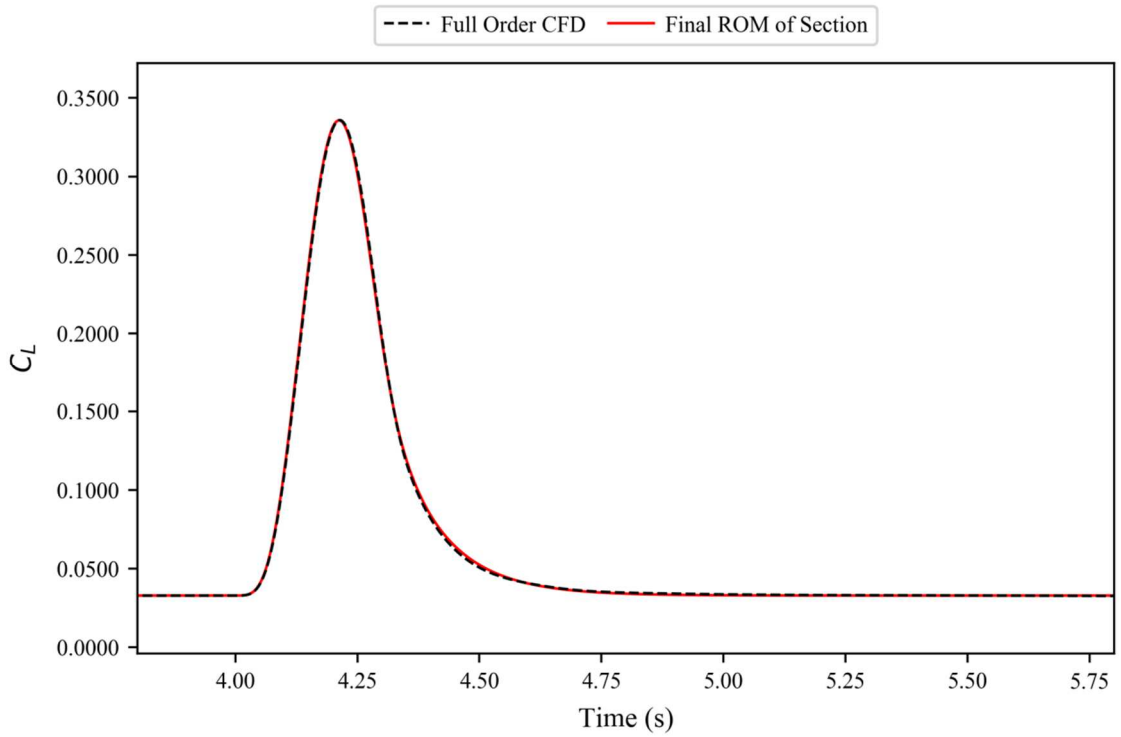

Figure 4.97. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 3, gust case 2.

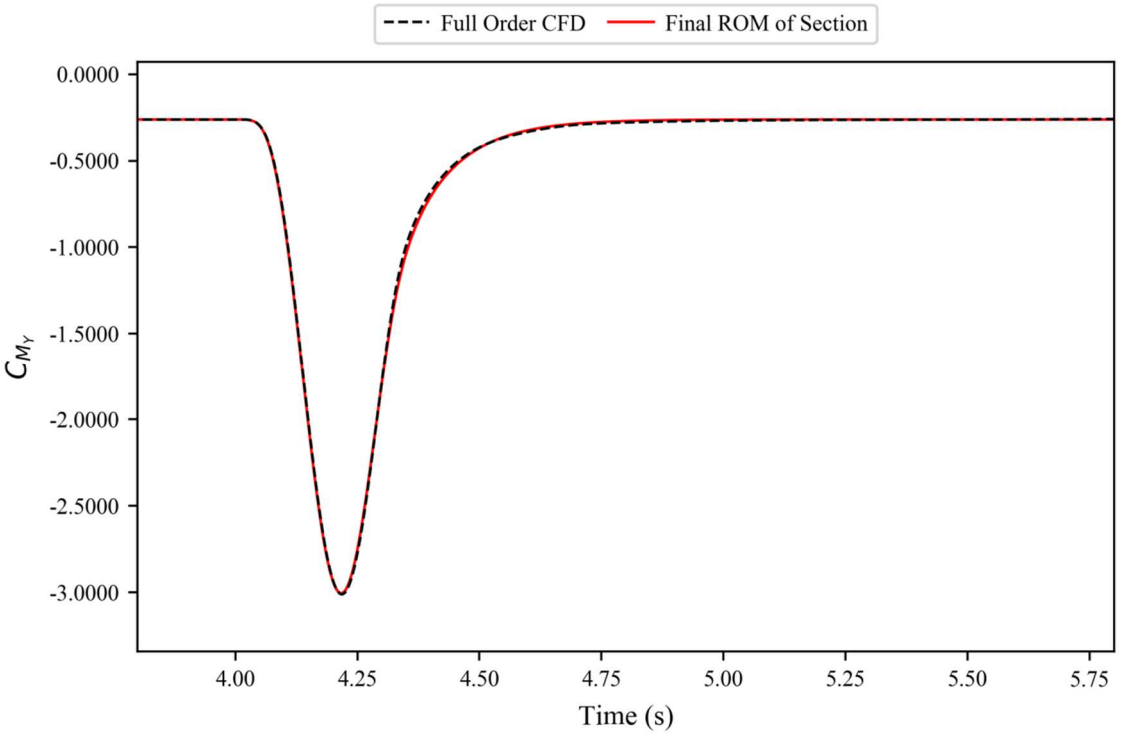

Figure 4.98. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 3, gust case 2.

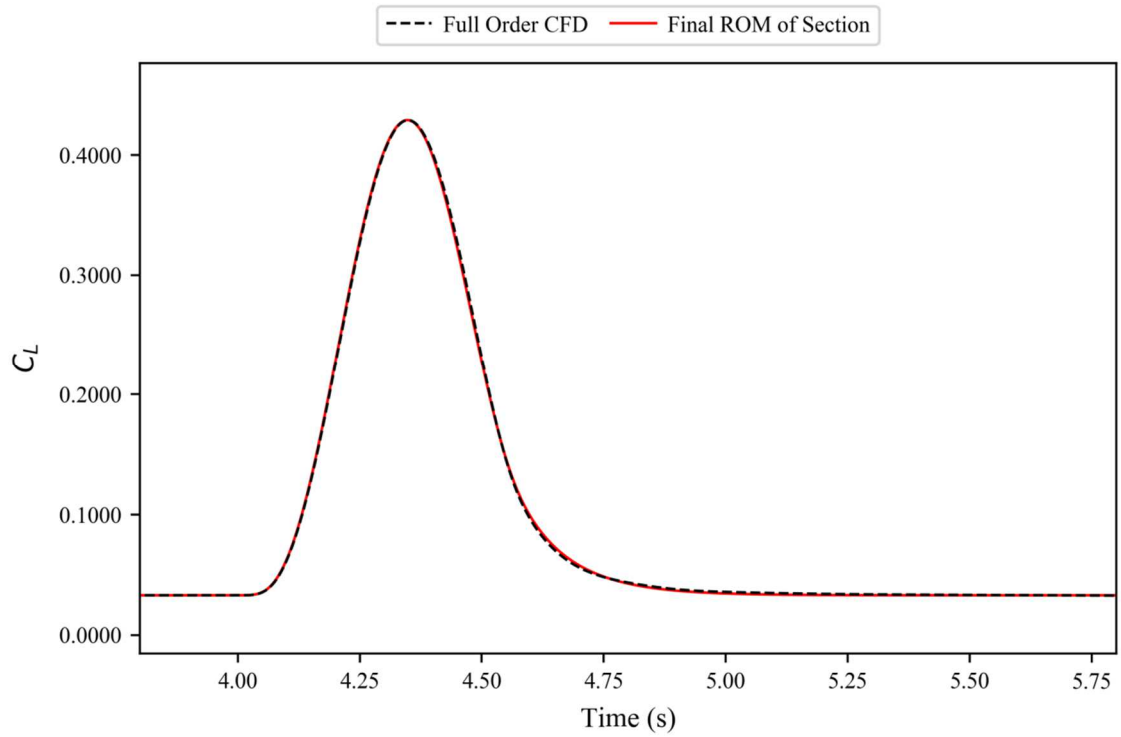

Figure 4.99. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 3, gust case 3.

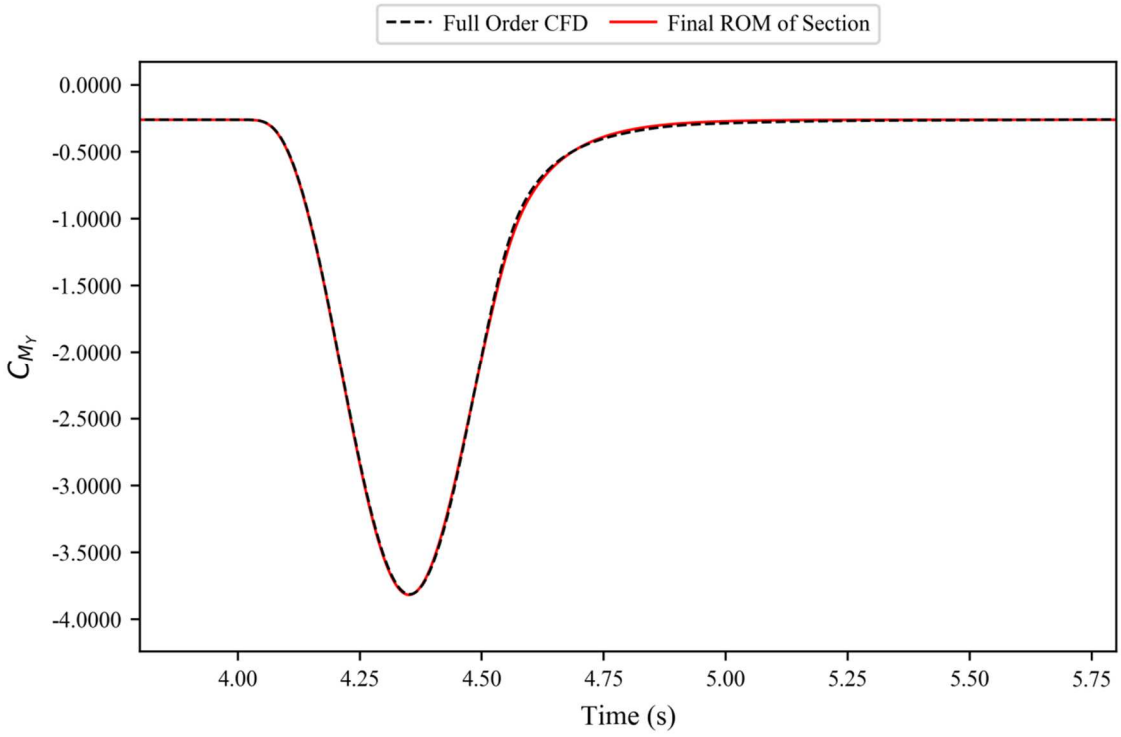

Figure 4.100. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 3, gust case 3.

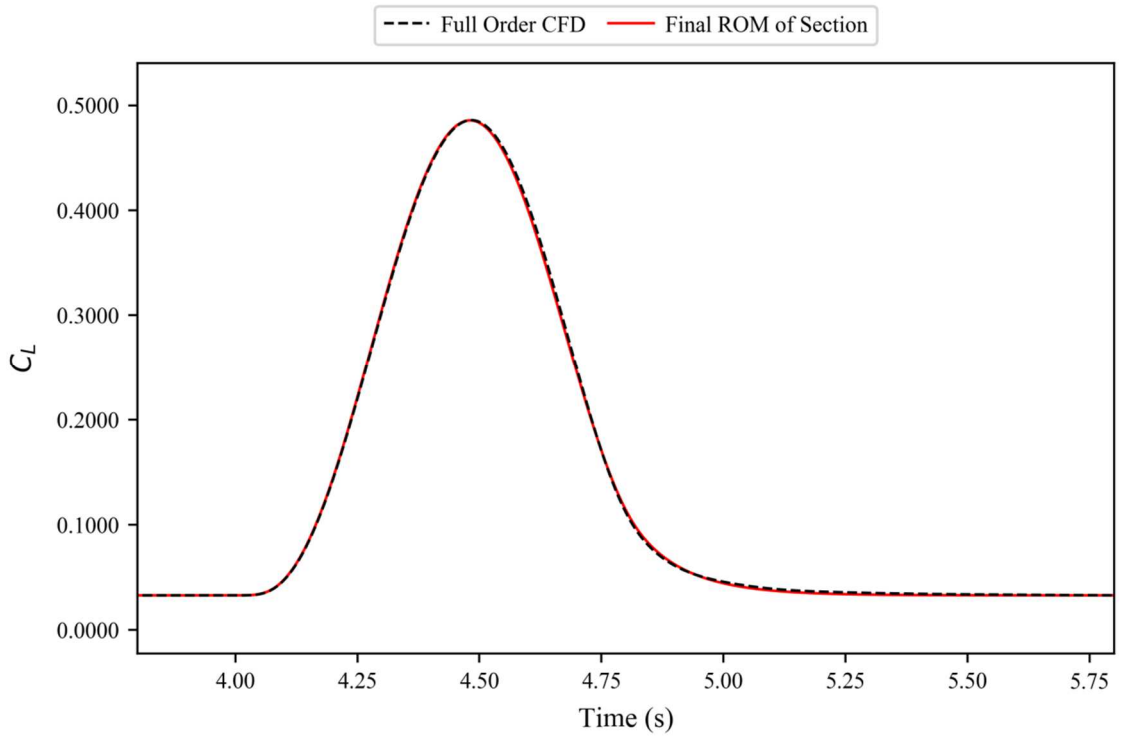

Figure 4.101. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 3, gust case 4.

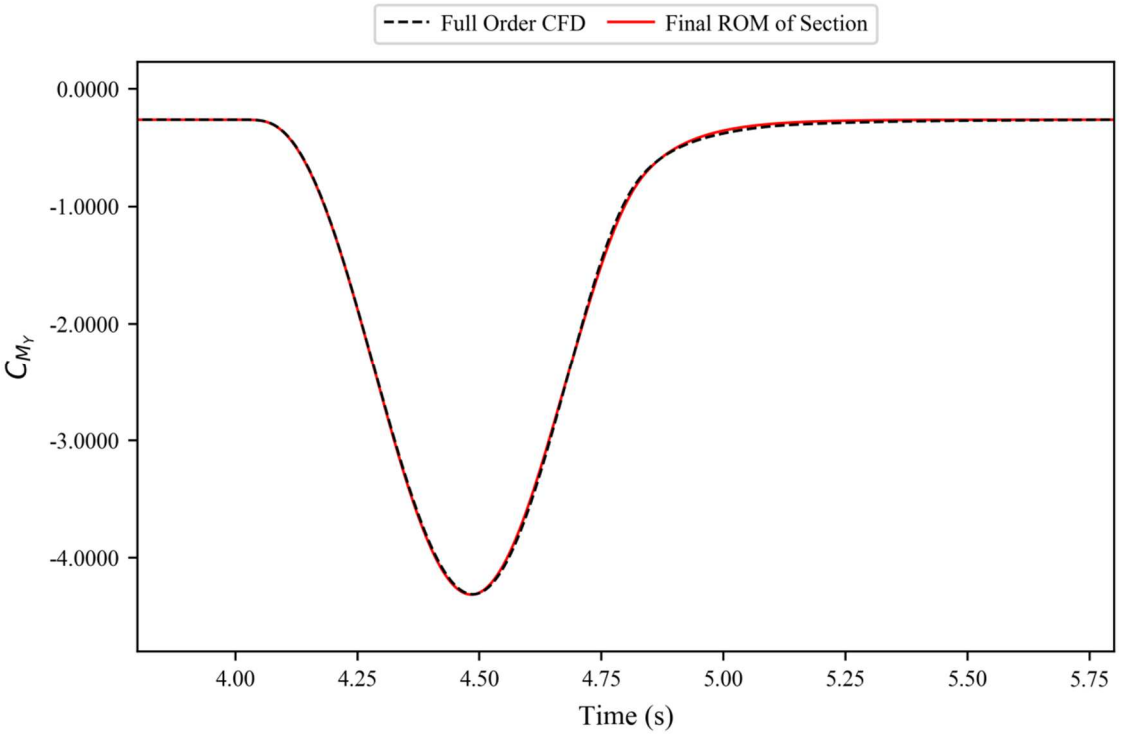

Figure 4.102. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 3, gust case 4.

4.6.3. Flight Point 4

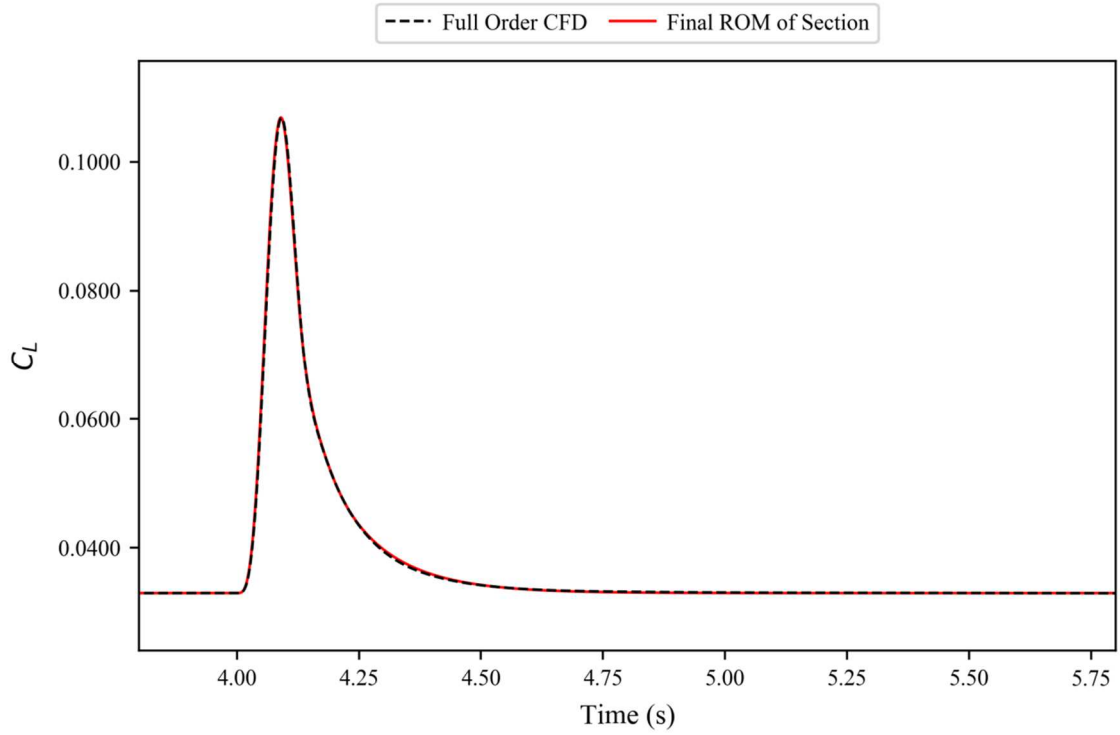

Figure 4.103. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 4, gust case 1.

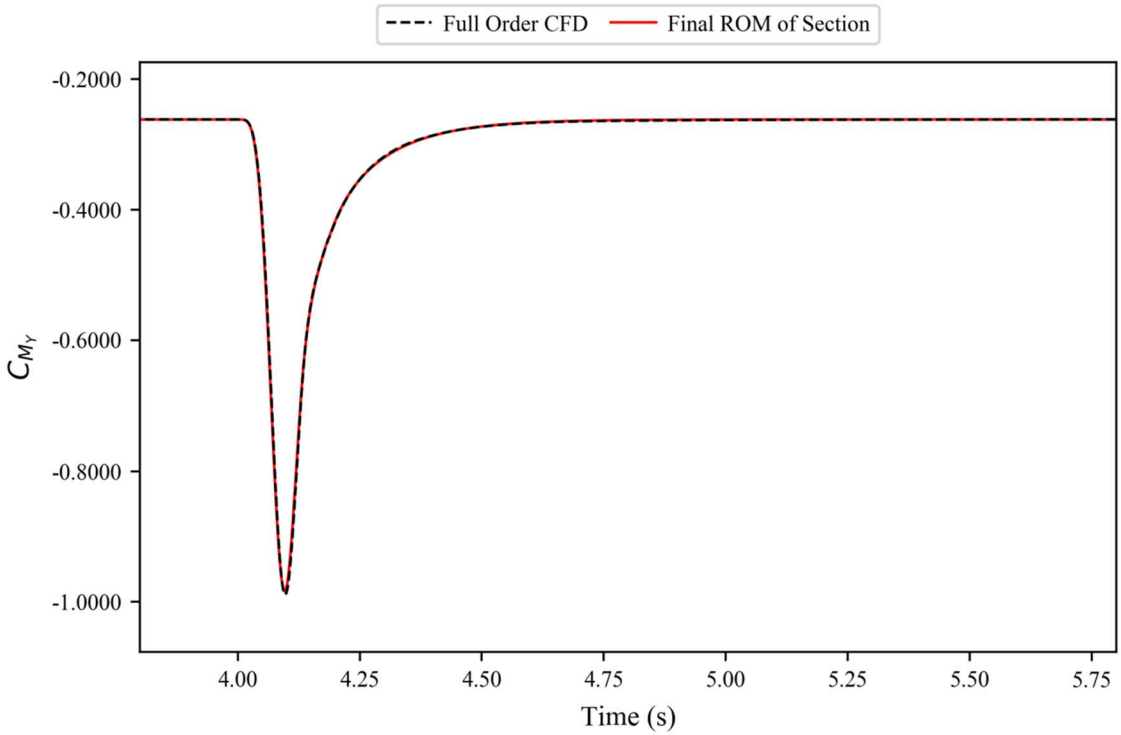

Figure 4.104. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 4, gust case 1.

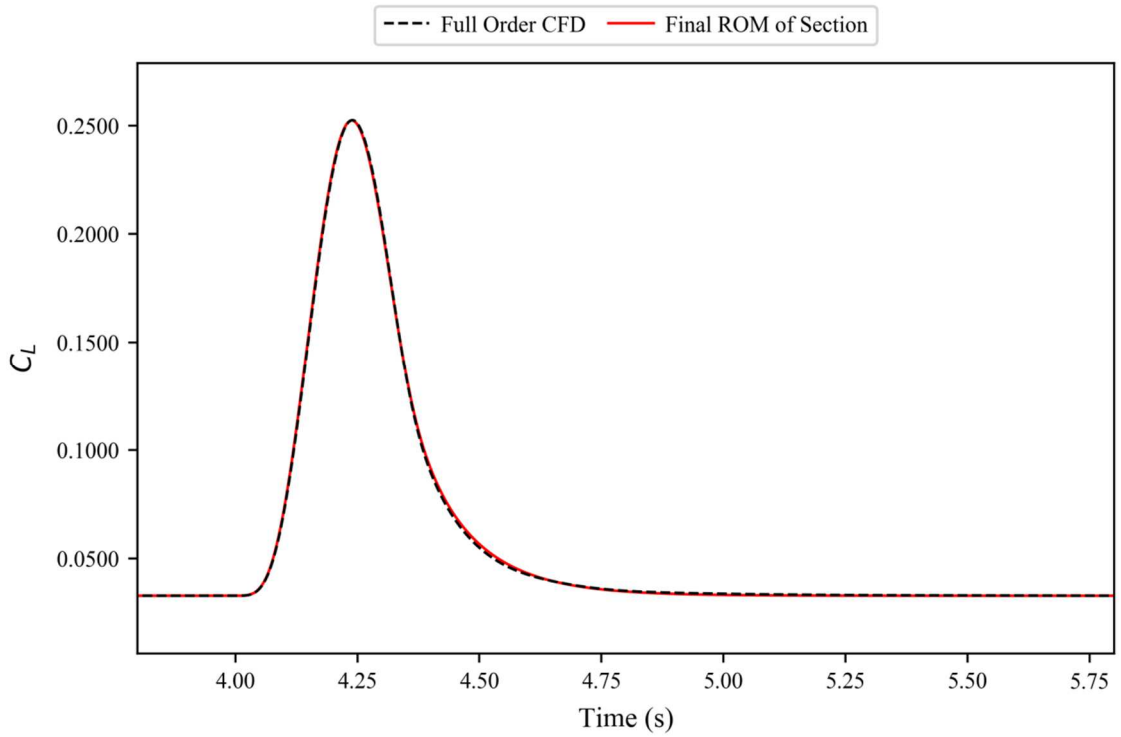

Figure 4.105. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 4, gust case 2.

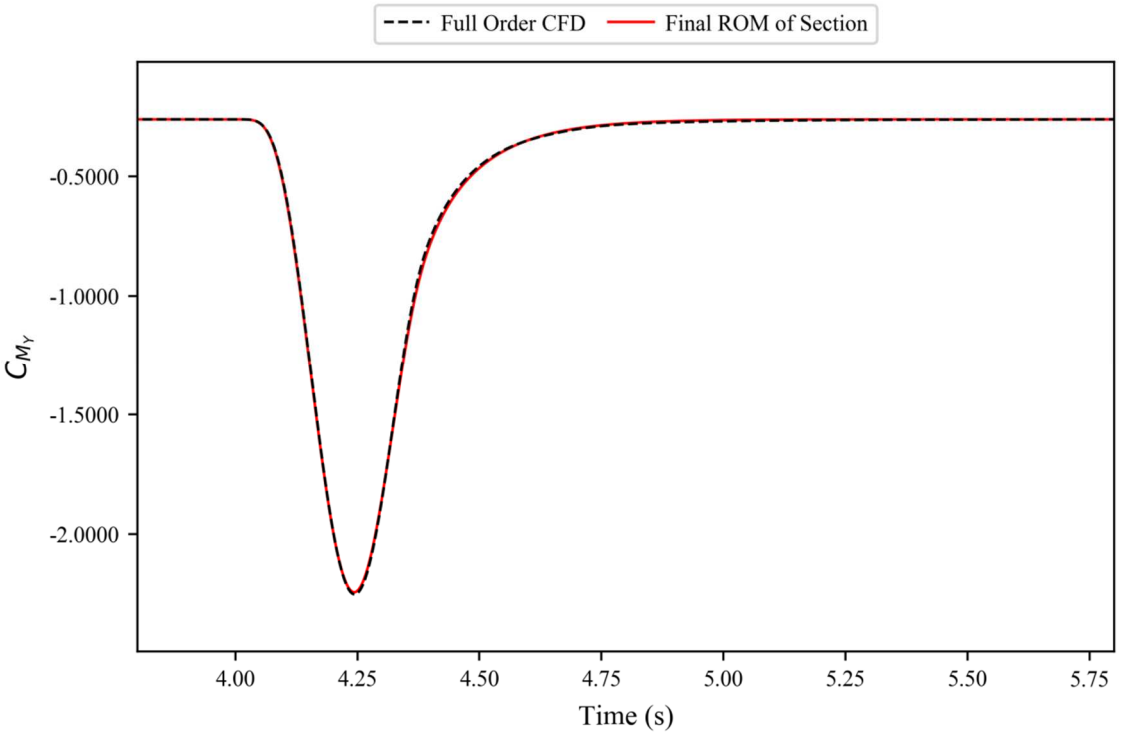

Figure 4.106. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 4, gust case 2.

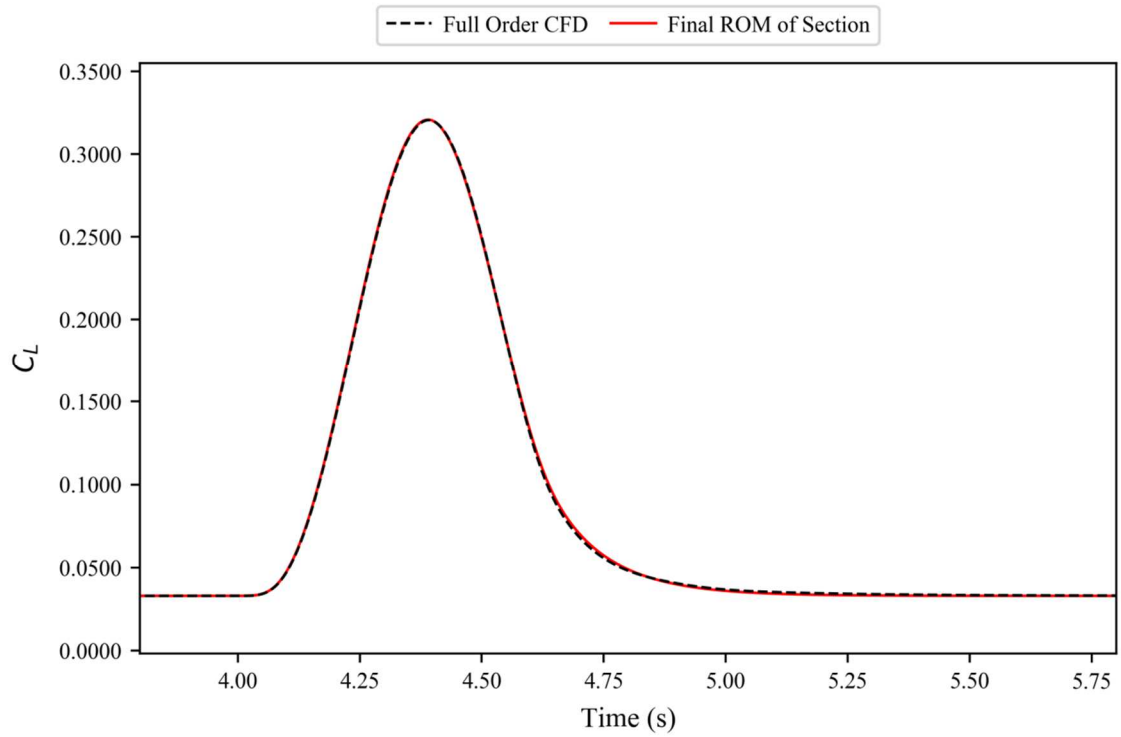

Figure 4.107. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 4, gust case 3.

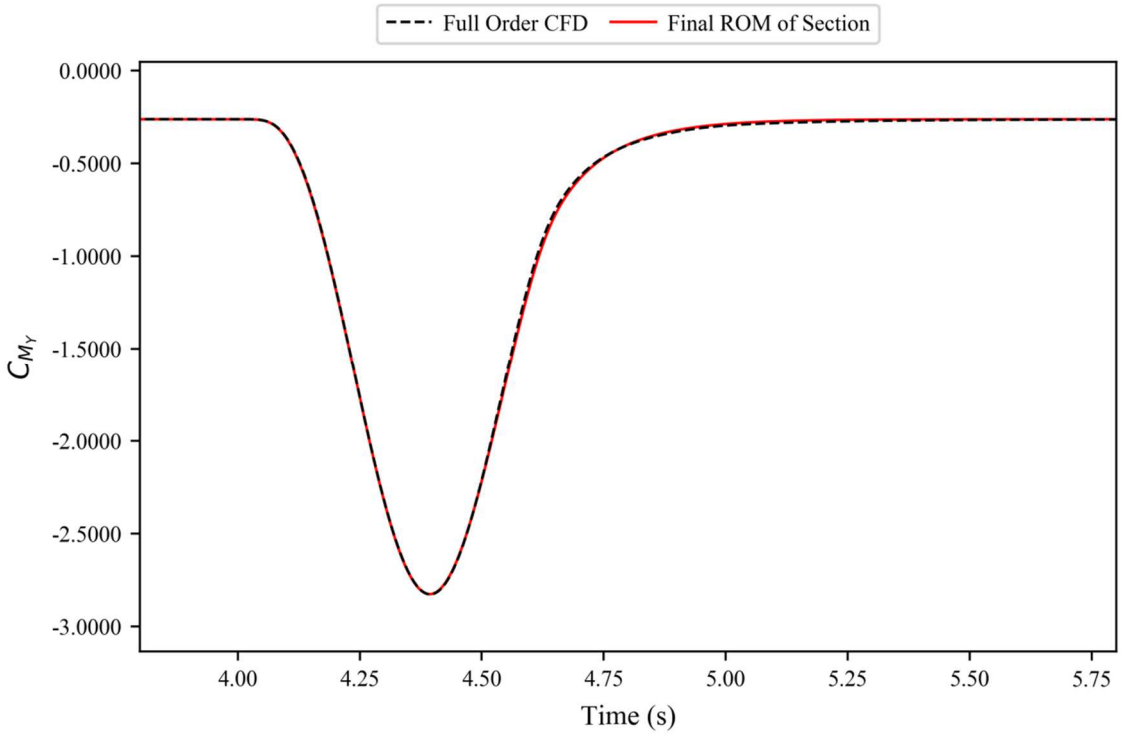

Figure 4.108. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 4, gust case 3.

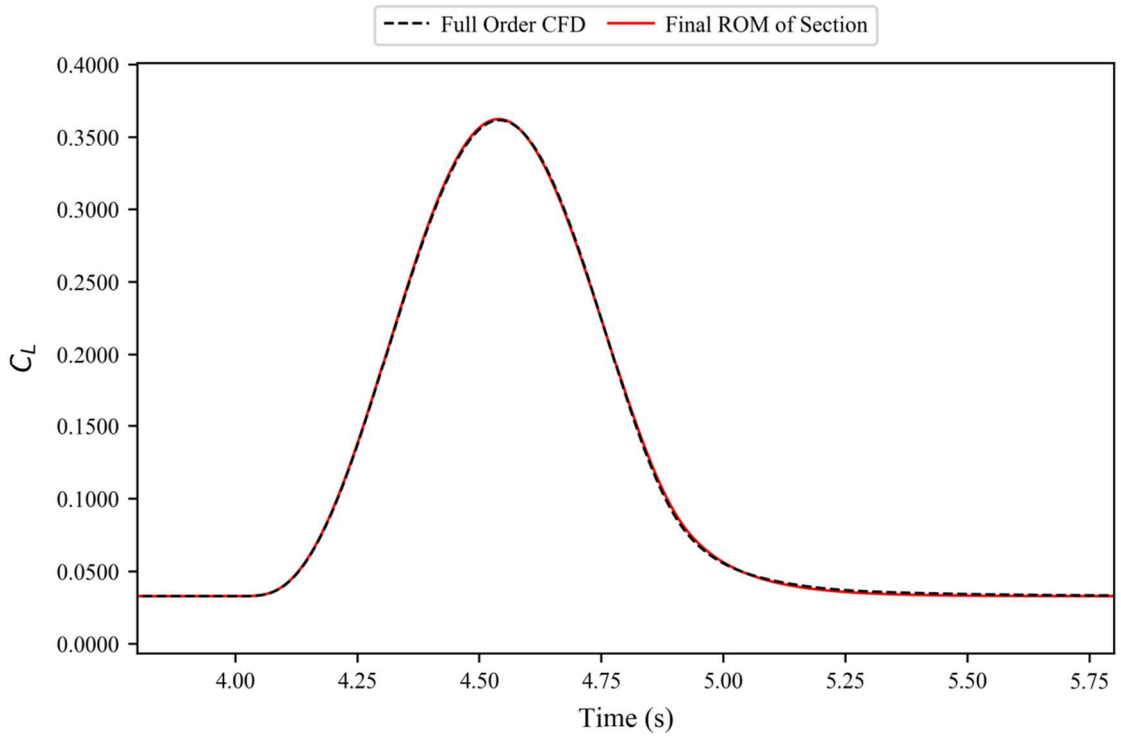

Figure 4.109. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 4, gust case 4.

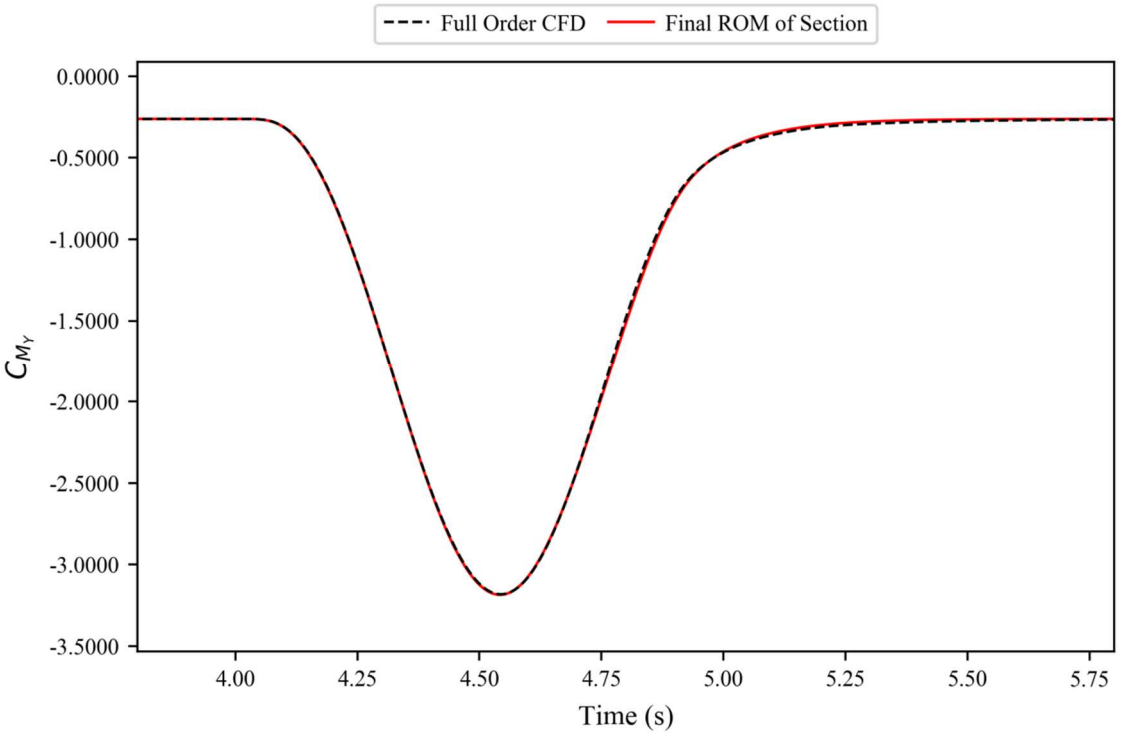

Figure 4.110. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 4, gust case 4.

4.6.4. Flight Point 5

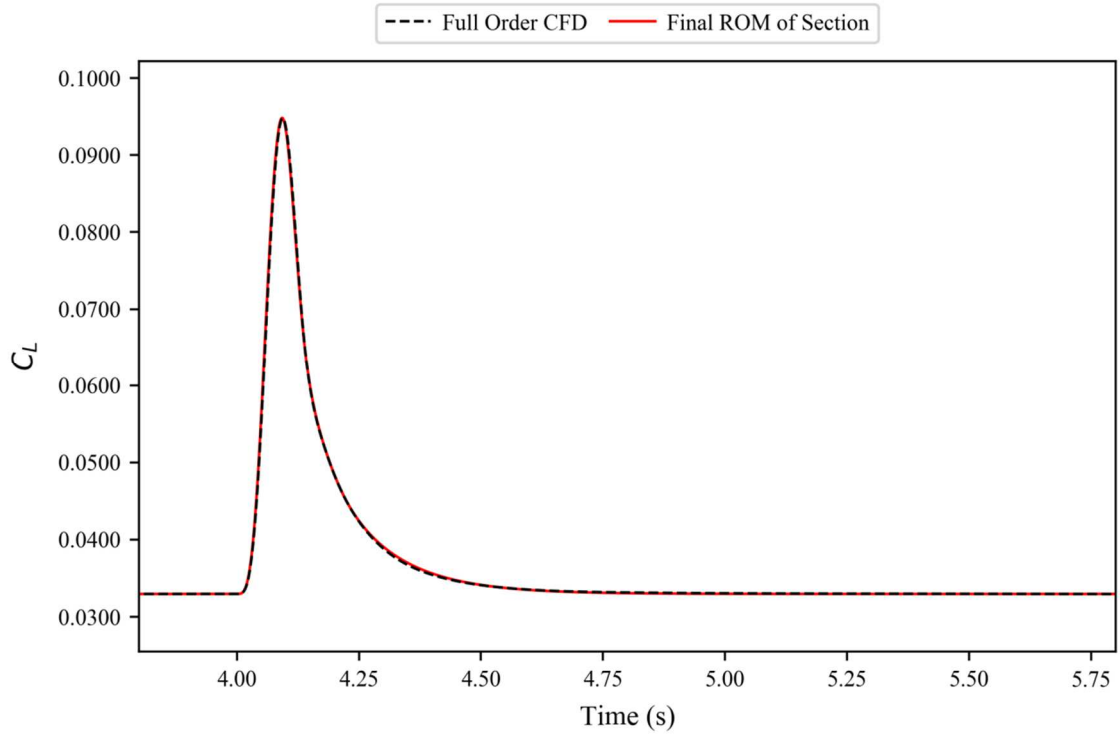

Figure 4.111. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 5, gust case 1.

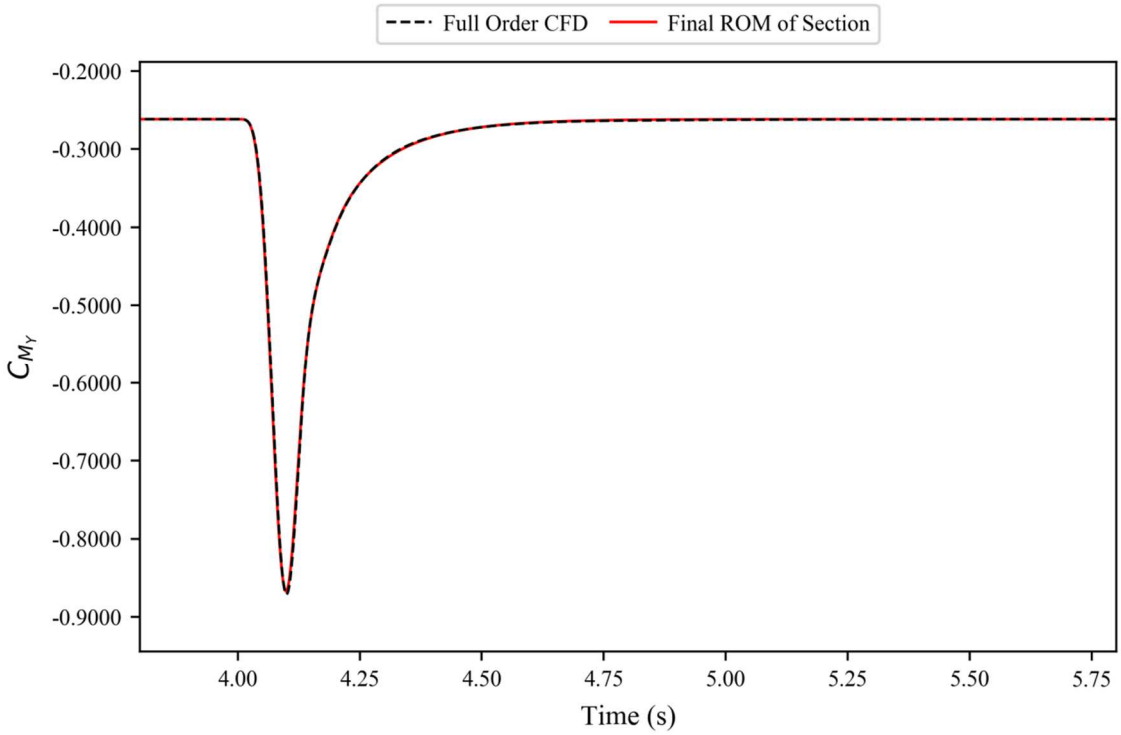

Figure 4.112. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 5, gust case 1.

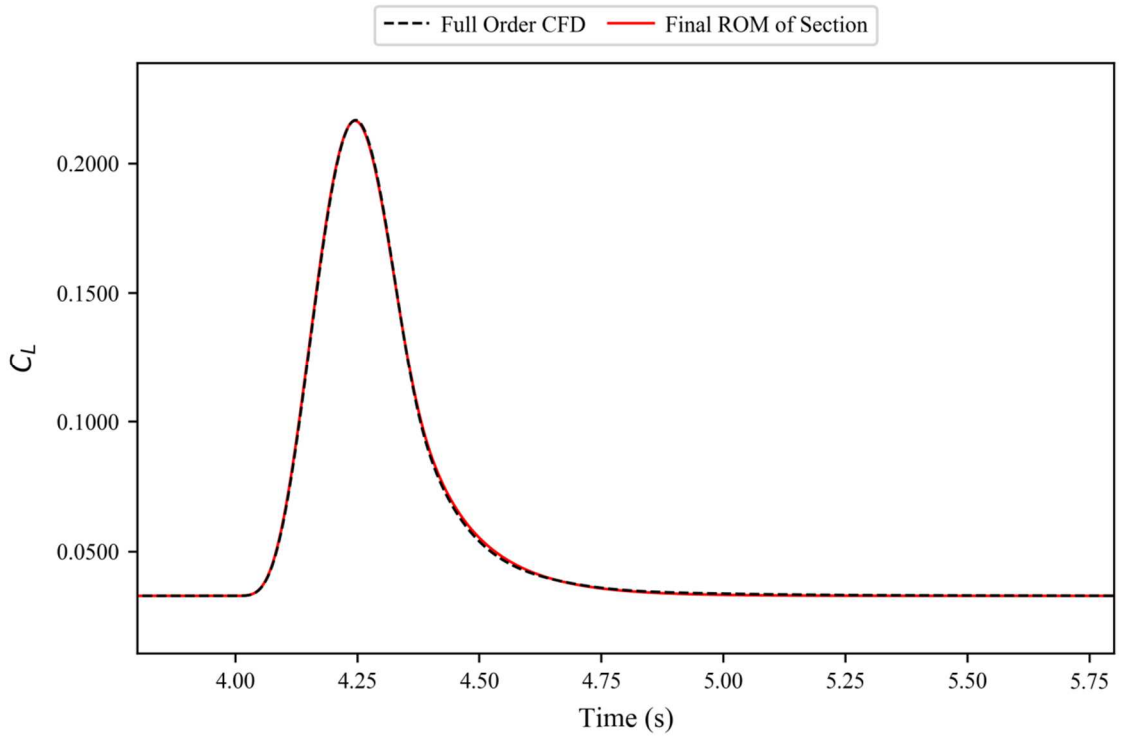

Figure 4.113. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 5, gust case 2.

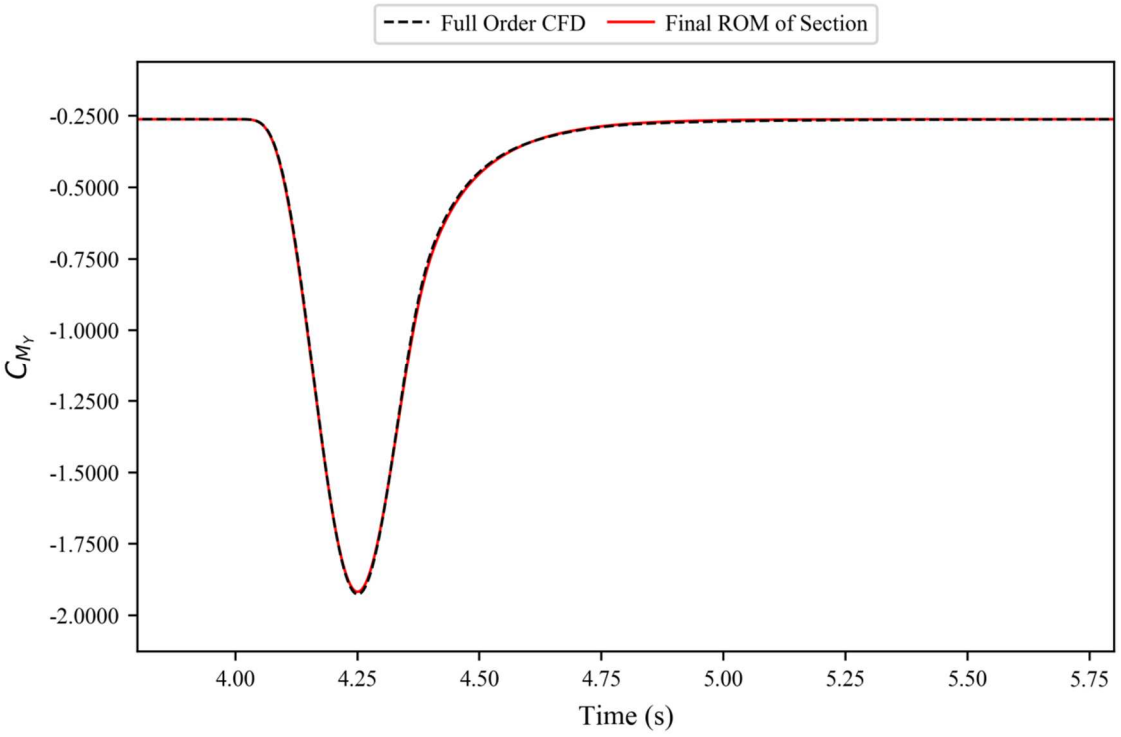

Figure 4.114. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 5, gust case 2.

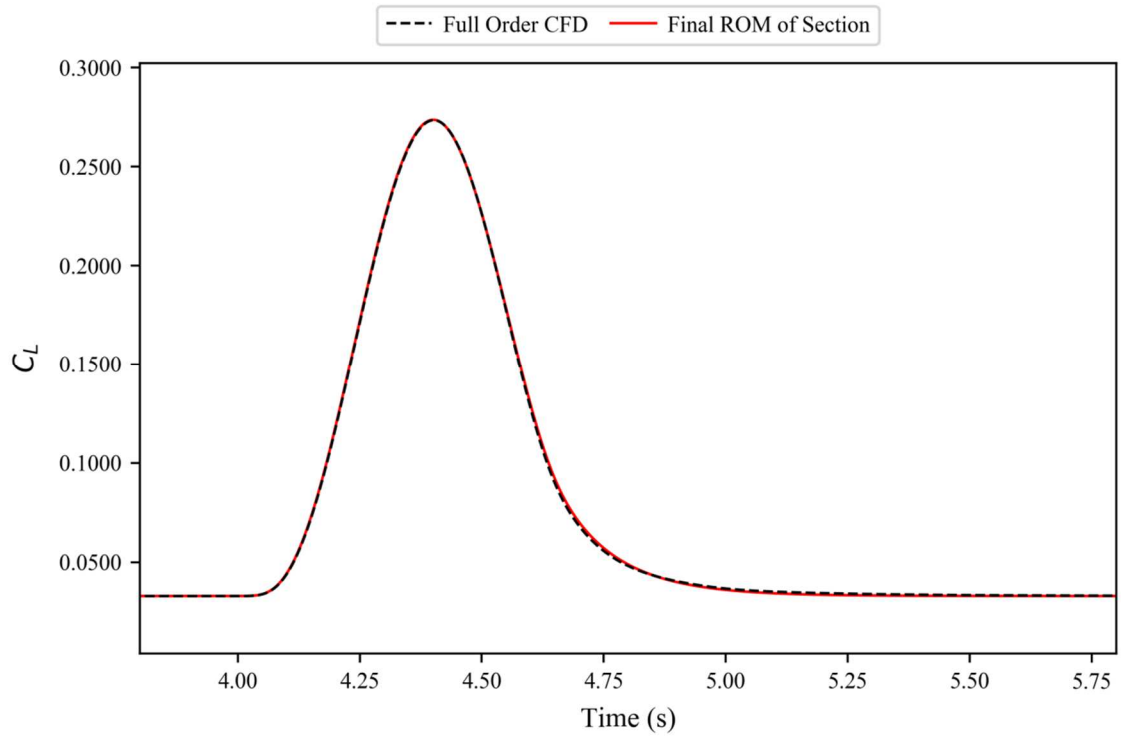

Figure 4.115. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 5, gust case 3.

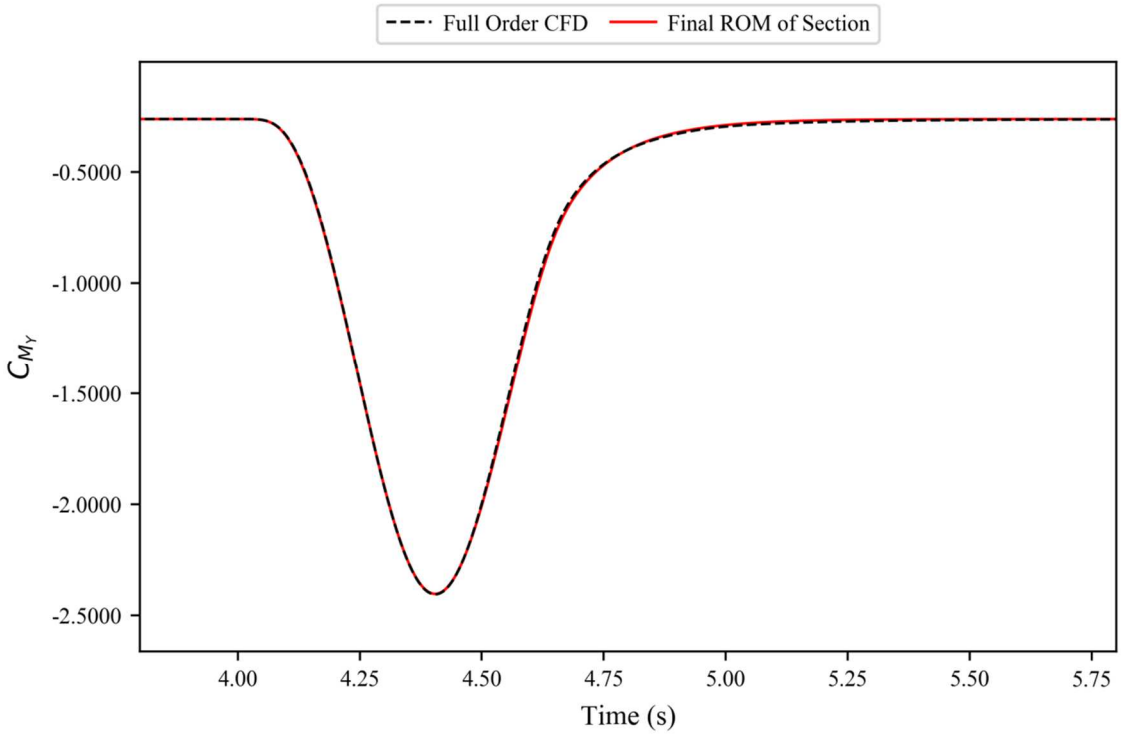

Figure 4.116. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 5, gust case 3.

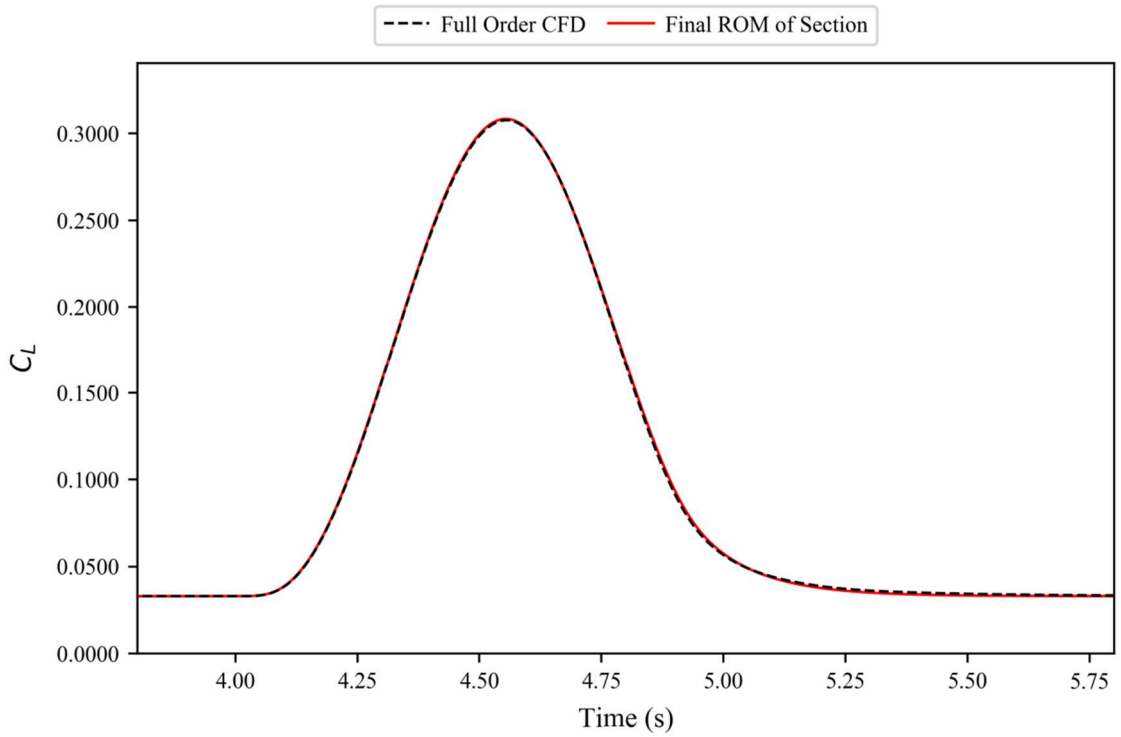

Figure 4.117. ROM results against full order CFD simulation for the coefficient of lift for the FFAST wing at flight point 5, gust case 4.

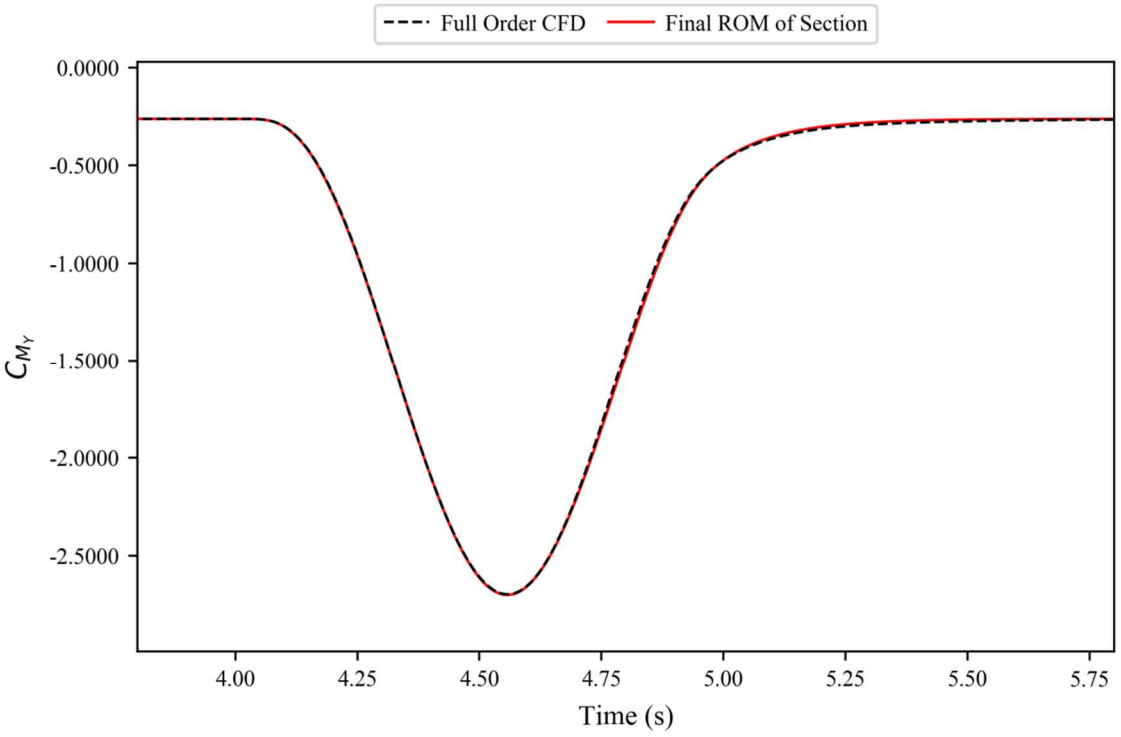

Figure 4.118. ROM results against full order CFD simulation for the coefficient of pitching moment for the FFAST wing at flight point 5, gust case 4.

4.6.5. Conclusions

All the results (Figures 4.59-4.66 and 4.87-4.118) show a very high level of accuracy; with the results often being almost indistinguishable from the full order CFD results. This highlights that the ROM method being taken forward has maintained, if not improved upon, the accuracy of the baseline ROM method. However, and most importantly, the ROM method being taken forward is far less computationally expensive compared to the baseline. Due to numerous setup parameters influencing the computational cost of running a CFD simulation and/or building a ROM, it is impossible to fully quantify these computational savings. However, it can be at least considered, based on the number of time steps used.

For flight point 2, the baseline ROM method required two sharp-edged gusts, each using 450 time steps post-impact and requiring a minimum of ~450 time steps to propagate the gust from outside the domain to the start of the model. Therefore, the baseline ROM method required 1,800+ time steps to be constructed. Again, for flight point 2, the final ROM method used 87 time steps post-impact and 7 time steps to propagate the gust from outside the domain to the start of the model. Therefore, the final ROM method in this chapter required 94 time steps to be constructed. Whilst not a perfect comparison (for example, the number of iterations per time step can, and typically does, vary), the fact that the final ROM method requires just 5.22% the number of time steps used by the baseline (for flight point 2) highlights the extremely large computational savings achieved. It is also worth noting that the baseline ROM method had a similar computational cost as a single '1-cosine' full order CFD simulation.

5. ROM Applied to Whole Aircraft Model

In Chapter 0, the developed ROMs were tested on a rigid wing in inviscid flow. To further test their capabilities, it is necessary to consider a more complex model and viscosity. In this chapter ROMs are derived from viscous CFD of a whole aircraft. As before, the author was not involved in the development of the model, and as such it would be distracting from the work carried out to go into too much detail; however, for completeness, a brief overview is provided.

This aircraft is a rigid, generic, wide bodied aircraft (i.e. an aircraft that is sufficiently large as to contain two aisles when used for commercial passenger flight). To reduce the computational cost, symmetry is exploited by only modelling half the aircraft. A plane of symmetry is used within the CFD solver to allow this half model to be simulated as if it was the whole aircraft (whilst halving the number of nodes and cells within the mesh); as shown in Figure 5.1.

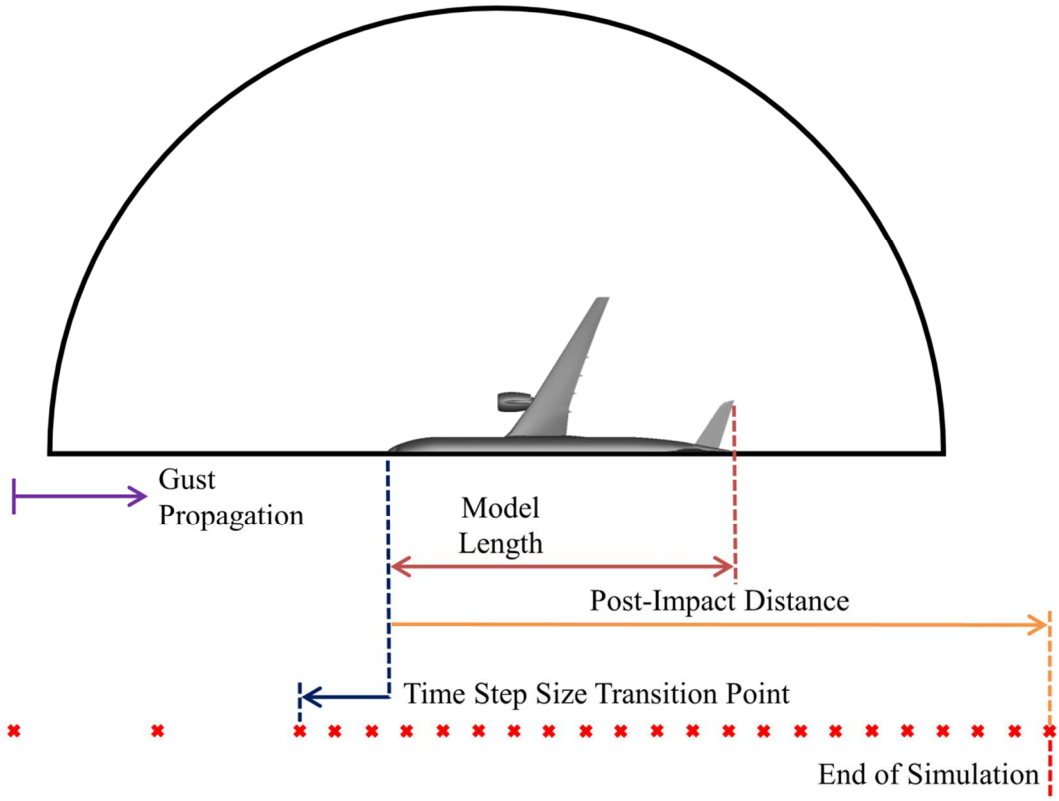

Figure 5.1. Representation (not to scale) of the top down view of the domain for the generic wide-bodied aircraft geometry.

The aerodynamic model consisted of an unstructured mesh throughout a hemisphere domain (see Figures 5.2 and 5.3). The mesh was made up of 2,459,947 hexahedron

cells, 60,538 prism cells, 59,025 square based pyramid cells and 2,640,552 tetrahedron cells; giving a total of 5,220,062 three-dimension cells. The mesh also had 58,036 triangular surface elements and 77,641 surface quadrilateral elements; giving a total of 135,677 surface elements.

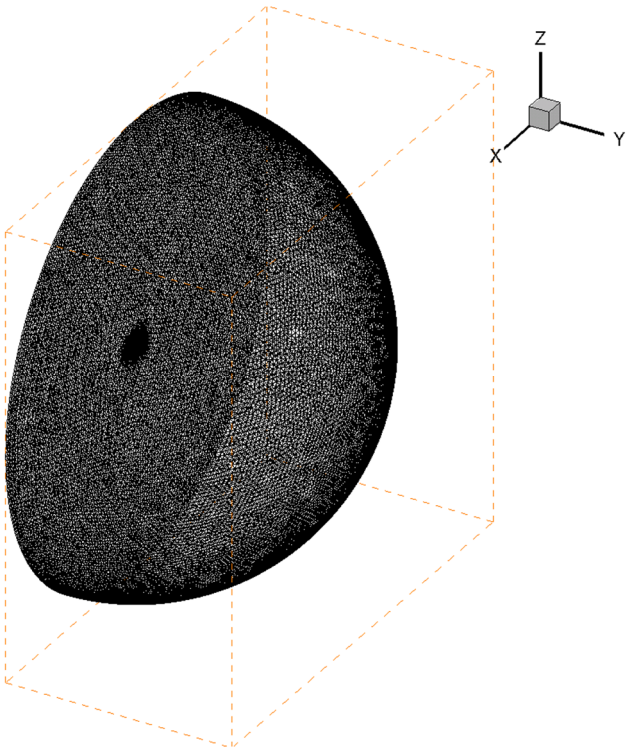

Figure 5.2. Domain mesh for the whole aircraft model.

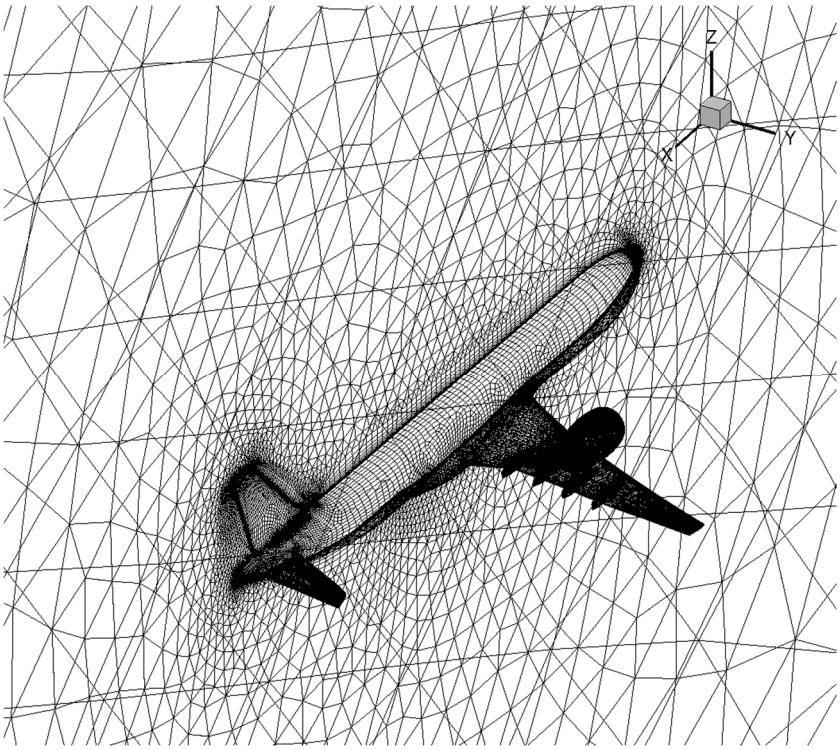

Figure 5.3. Surface mesh for the whole aircraft model.

For these viscous simulations, the FANS equations are solved for (see Section 2.2.3), with the SA-noft2 being used as the turbulence model (see Section 2.2.4). Additionally, all five flight points (see Table 5) were tested before the unsteady cases were carried out to check that the y+ value was suitably small. As can be seen in Table 4, for all cases the maximum value of y+ was below 2 which, whilst not ideal, is unlikely to have caused any notable degradation in boundary layer modelling accuracy. Additionally, the y+ values across the surface of the model (see Figures 5.4 and 5.5) for the worst case (Flight Point 1) show that the maximum value is localised to a very small region toward the nose of the aircraft; with the rest of the model (critically, including the wing) appearing to have y+ values under 1. Therefore it was felt the boundary layer mesh was sufficiently fine for the cases being explored.

Table 4. y+ values.

Flight Point	**1**	**2**	**3**	**4**	**5**
Minimum y+	1.506×10^{-2}	1.506×10^{-2}	1.506×10^{-2}	1.506×10^{-2}	1.506×10^{-3}
Maximum y+	1.506×10^{0}	1.506×10^{0}	1.506×10^{0}	1.506×10^{-1}	1.506×10^{-1}

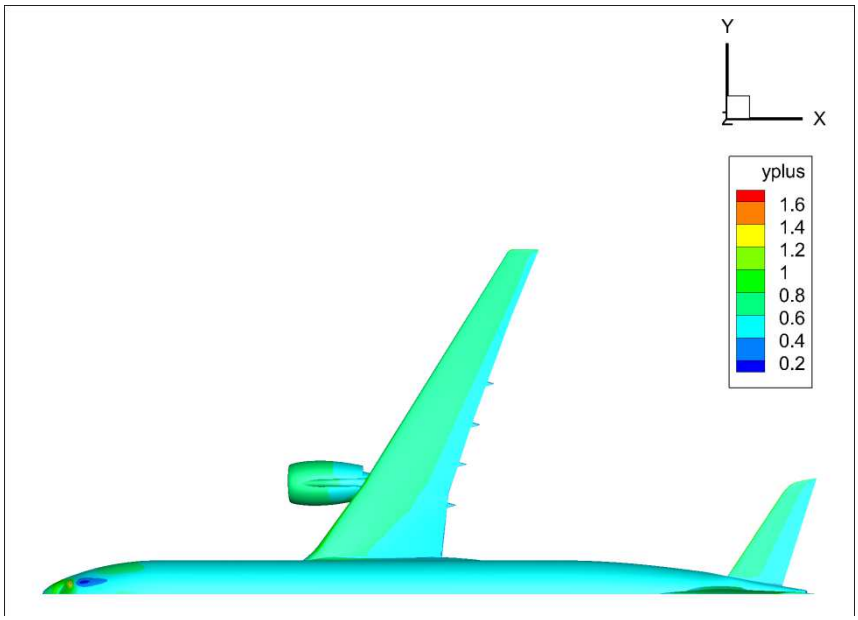

Figure 5.4. y+ values on the top surface of the generic wide-bodied aircraft, for Flight Point 1.

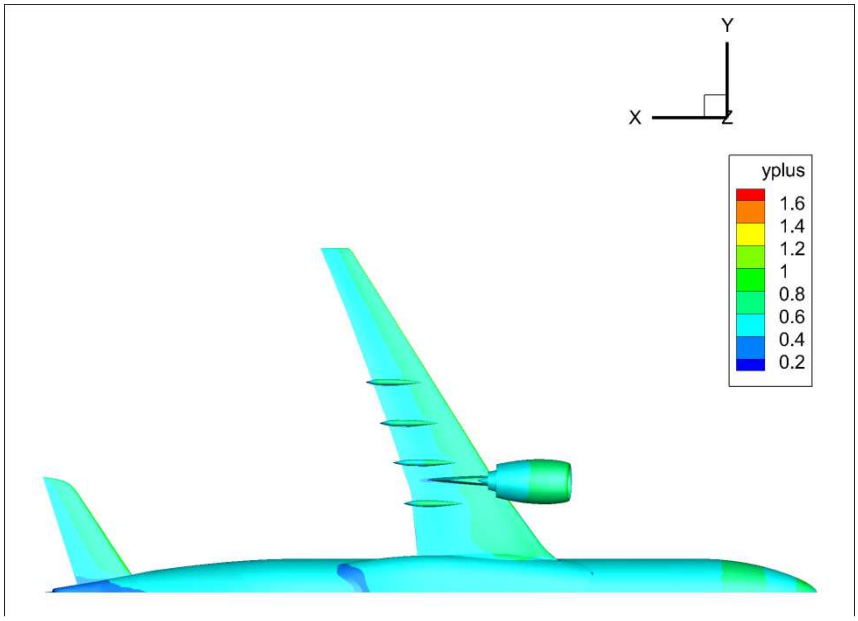

Figure 5.5. y+ values on the top surface of the generic wide-bodied aircraft, for Flight Point 1.

For this model, a gust alleviation factor was calculated in accordance with CS-25 regulations [2] to be equal to 0.7786. This alleviation factor is used (again, in accordance to the CS-25 regulations) in the calculation of the peak gust velocities. Similarly to Chapter 0, five flight points were selected, each with four gust cases; as seen in Table 5. Due to commercial sensitivity, all results in this chapter are normalised by the absolute value of either the maximum (Cl) or minimum (Cm) value from the CFD results.

Table 5. Details of gusts explored for the generic, wide-bodied aircraft model.

Flight Point	Mach Number	Altitude (ft)	Altitude (m)	Gust 1 18m Velocity (ms^{-1})	Gust 2 80m Velocity (ms^{-1})	Gust 3 146m Velocity (ms^{-1})	Gust 4 214m Velocity (ms^{-1})
1	0.500	0	0	8.798	11.281	12.470	13.291
2	0.735	0	0	8.798	*	12.470	13.291
3	0.800	0	0	8.798	11.281	12.470	13.291
4	0.800	29,9995	9,142.476	5.700	7.309	8.080	8.612
5	0.800	43,000	13,106.400	4.650	5.963	6.592	7.026

** Due to a technical issue, the CFD solution for Gust 2 at Flight Point 2 could not be obtained; as such this case is not included in the results presented.*

5.1. Variable Time-Step Based ROM

Before implementing the step-down ROM method put forward in Sections 4.4 and 4.5.1, the effect that the transition from large time steps to small ones (within the sharp-edged gust) has on ROM accuracy is again checked, for this viscous full aircraft case, to verify that the previous conclusions are still valid. Previously (in Sections 4.3.2) 0.2 MAC lengths ahead of the model was deemed sufficient for the time step size transition to have no notable impact on the results; but it is possible that viscosity may have an impact on this.

5.1.1. Results

To test the ROM method, the gusts from flight point 2 for the generic wide-bodied aircraft are modelled. In all cases a ROM-size of 20 was used.

Figures 5.6-5.11 show very little deviation in the results for the different transition points; and as noted earlier in the thesis, this can make the results difficult to distinguish from one another. Therefore, it is necessary to examine the peak responses in more detail.

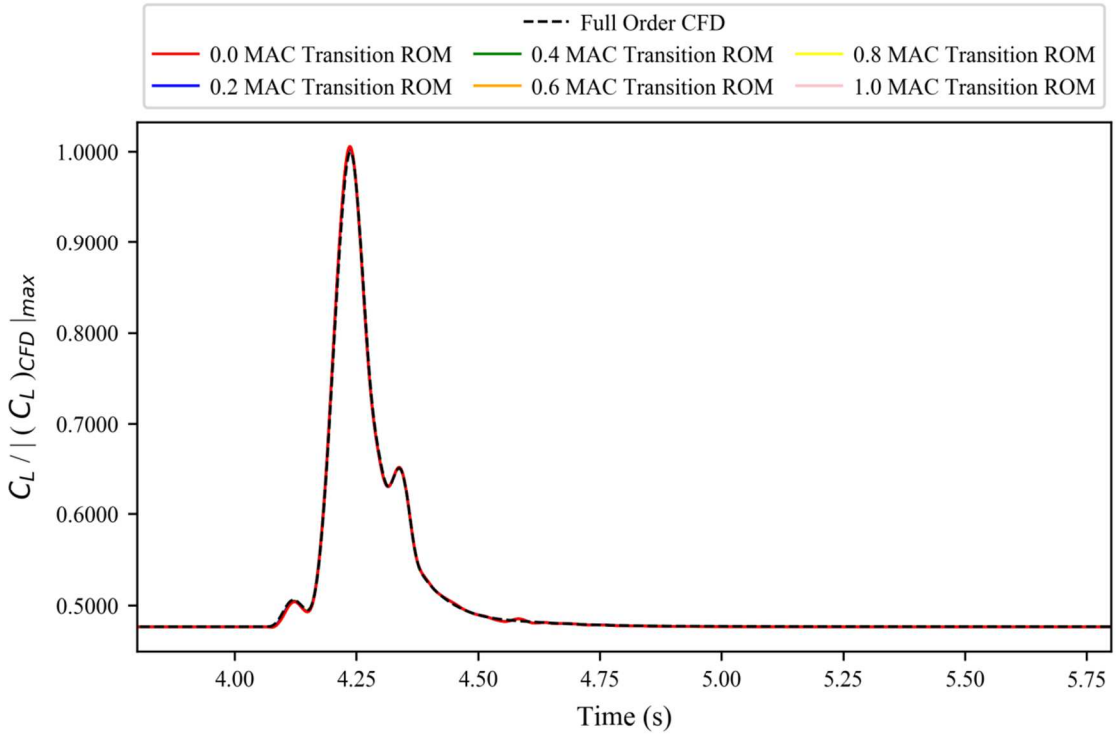

Figure 5.6. Variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 1.

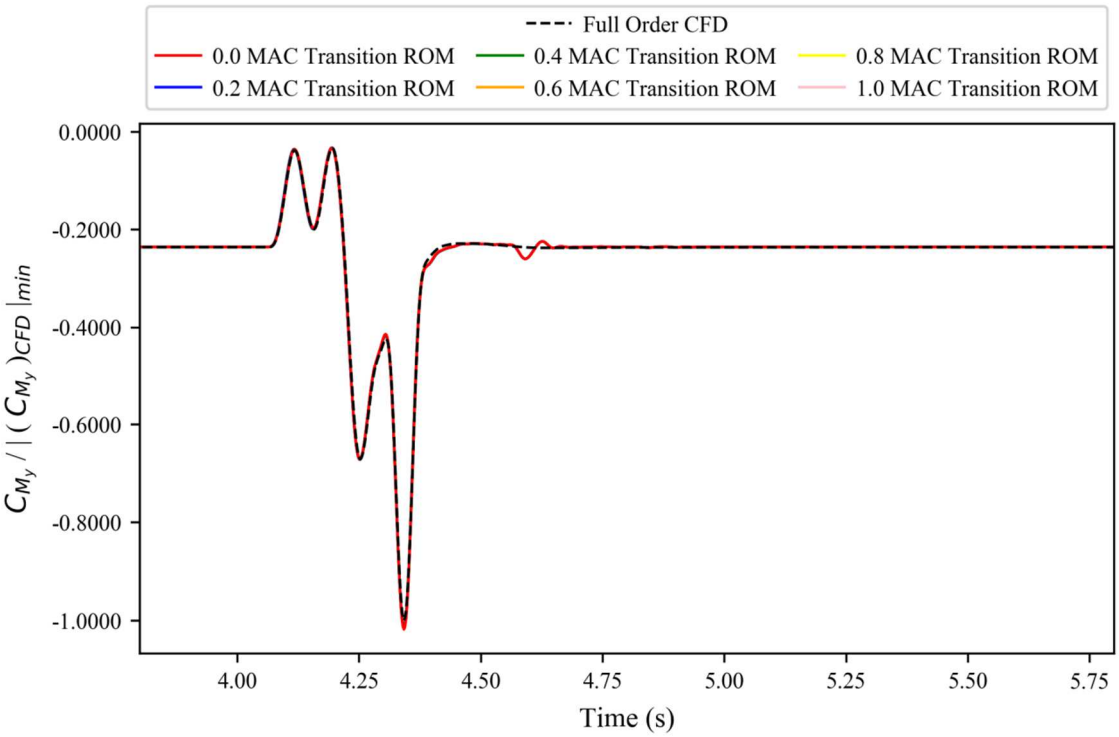

Figure 5.7. Variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 1.

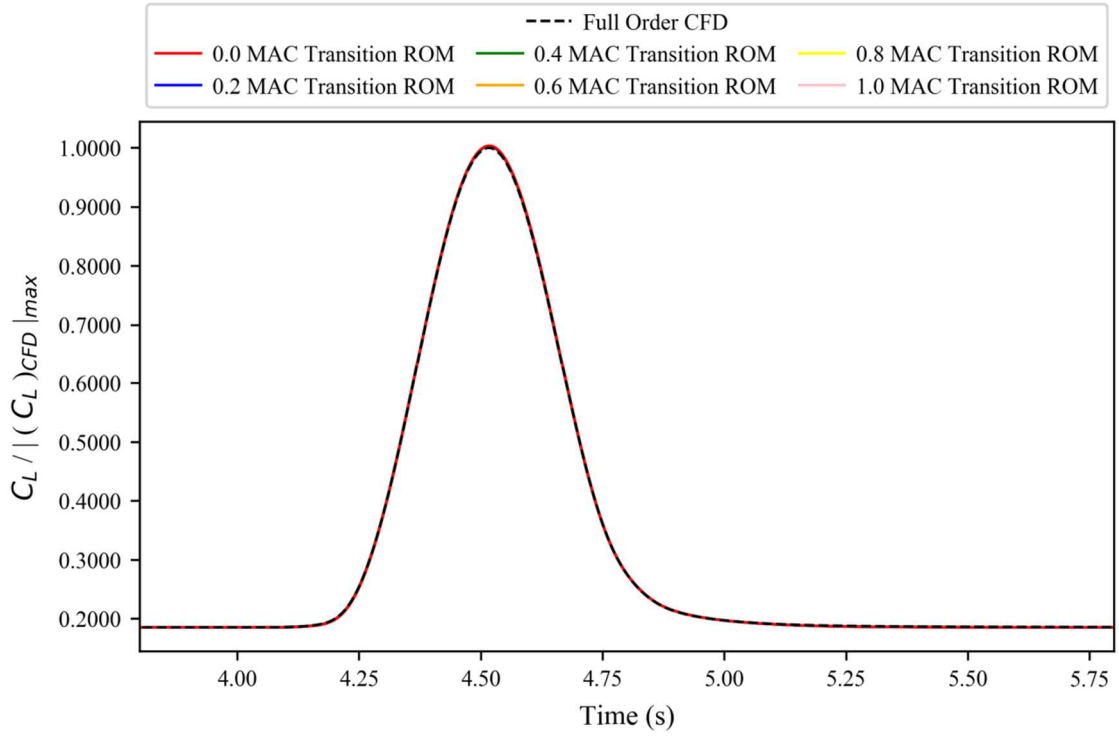

Figure 5.8. Variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 3.

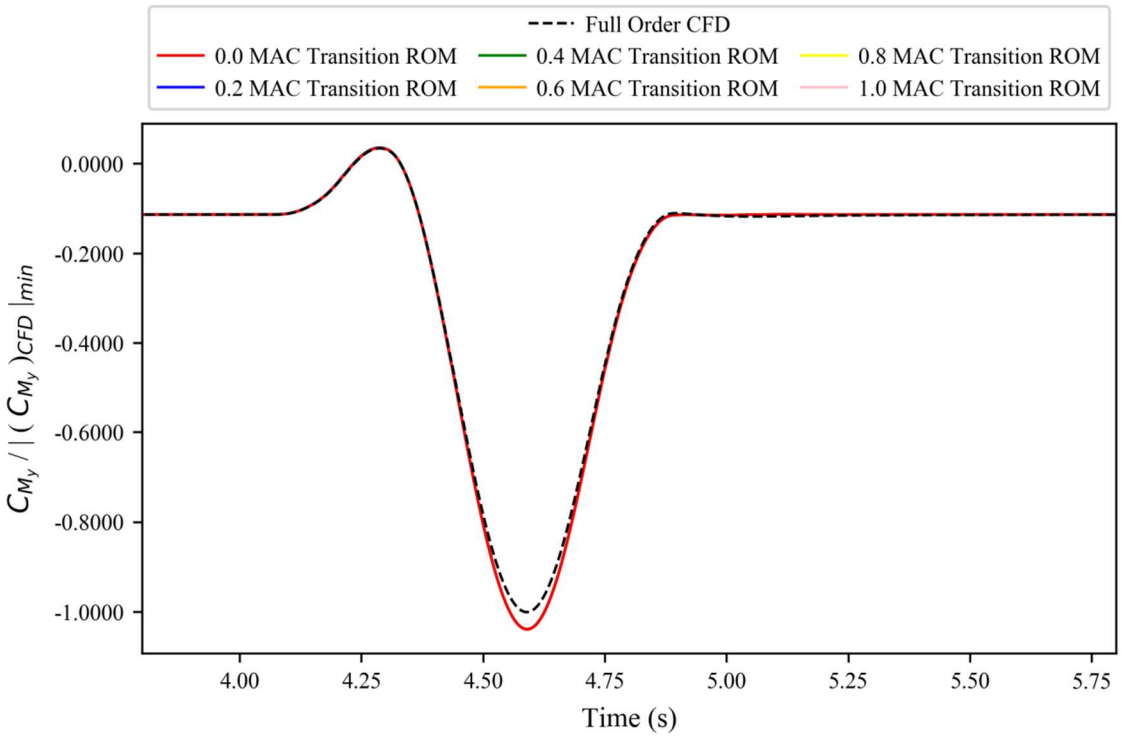

Figure 5.9. Variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 3.

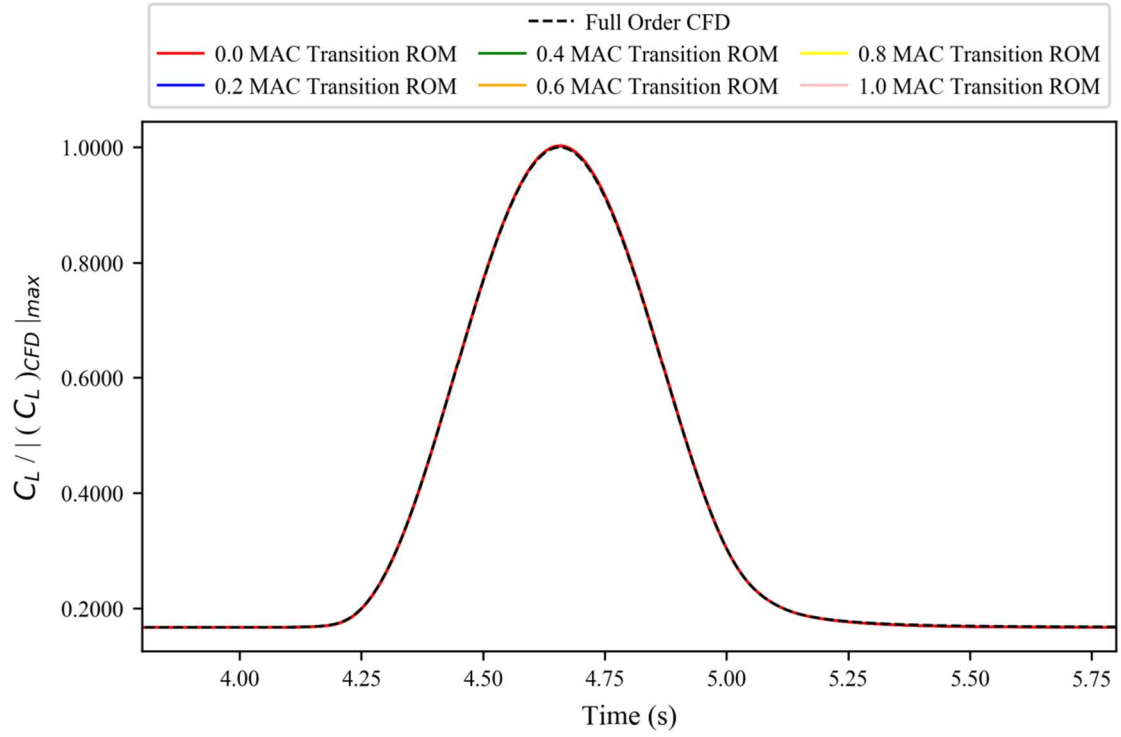

Figure 5.10. Variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 4.

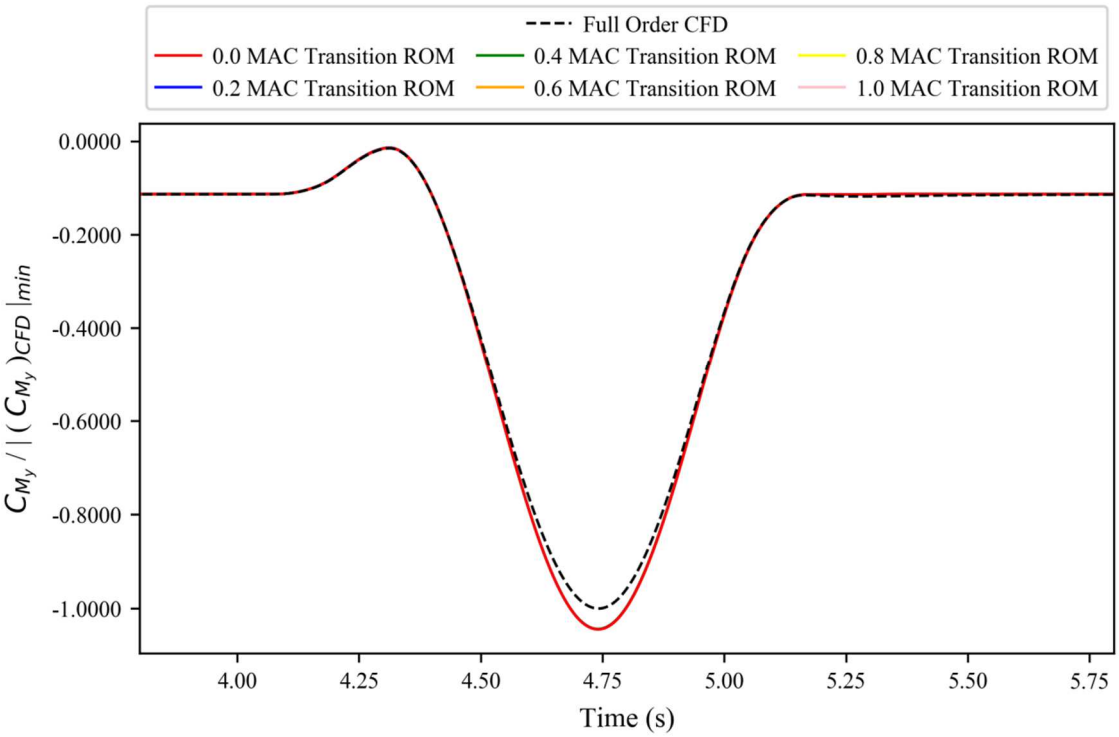

Figure 5.11. Variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 4.

Inspecting the peak results in Figures 5.12-5.16, it is immediately apparent that for the generic wide bodied aircraft model the location of the transition point, from large to small time steps, does not have any significant impact on the results; indeed the results are similar enough to still be hard to distinguish between.

Whilst the inclusion of viscosity might lead to some aspects of the system response being contained just ahead of the model, this is likely to be relatively negligible as most boundary layer effects (with the notable exception of flow separation) are carried downstream. However, a bigger difference between this model and the previously examined wing model, is the inclusion of a structure that is relatively neutral from an aerodynamic standpoint; the fuselage. As the primary source of the aircraft's response to a gust will occur due to the wing, it is perhaps unsurprising that there would not be any particularly important system behaviour captured shortly before the start of the fuselage (multiple meters ahead of the start of the wing).

Ideally, an additional study would have been carried out using the wing model from the previous chapter (Chapter 0) but with viscous flow. However, to do so would have required computational resources that were needed for either this full aircraft model, or the simpler aircraft model that includes flight mechanics (Chapter 0), and as a result of

the industrial sponsorship of this PhD, the decision was made to look at the more industrial relevant test cases.

As viscous effects at the nose of the fuselage are expected to be negligible, it would be reasonable to conclude that the transition point should be located at the start of the model. However, due to how the additional 0.2 MAC lengths typically only adds ~3 time steps, along with the fact that this made a notable difference for the wing model (see Sections 4.3.1 and 4.3.2) the previous choice of 0.2 MAC lengths ahead of the model remains a suitable choice as it offers a good balance between computational savings and ROM robustness when applying it to different model geometries.

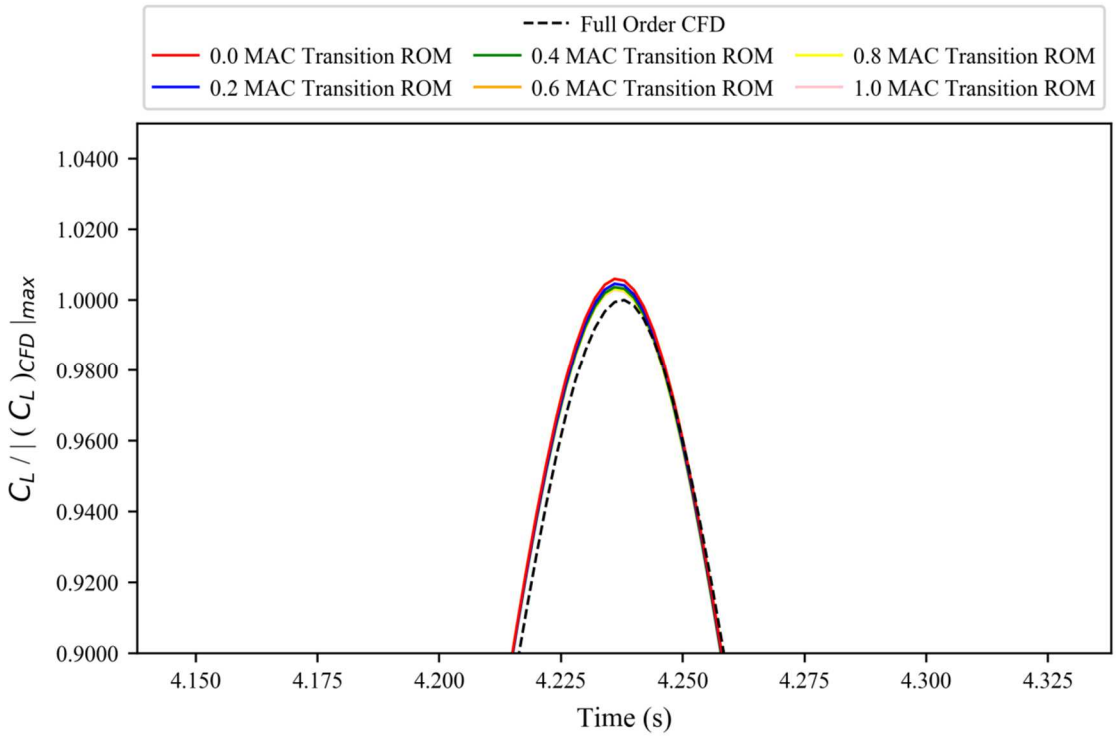

Figure 5.12. Peak response for the variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 1.

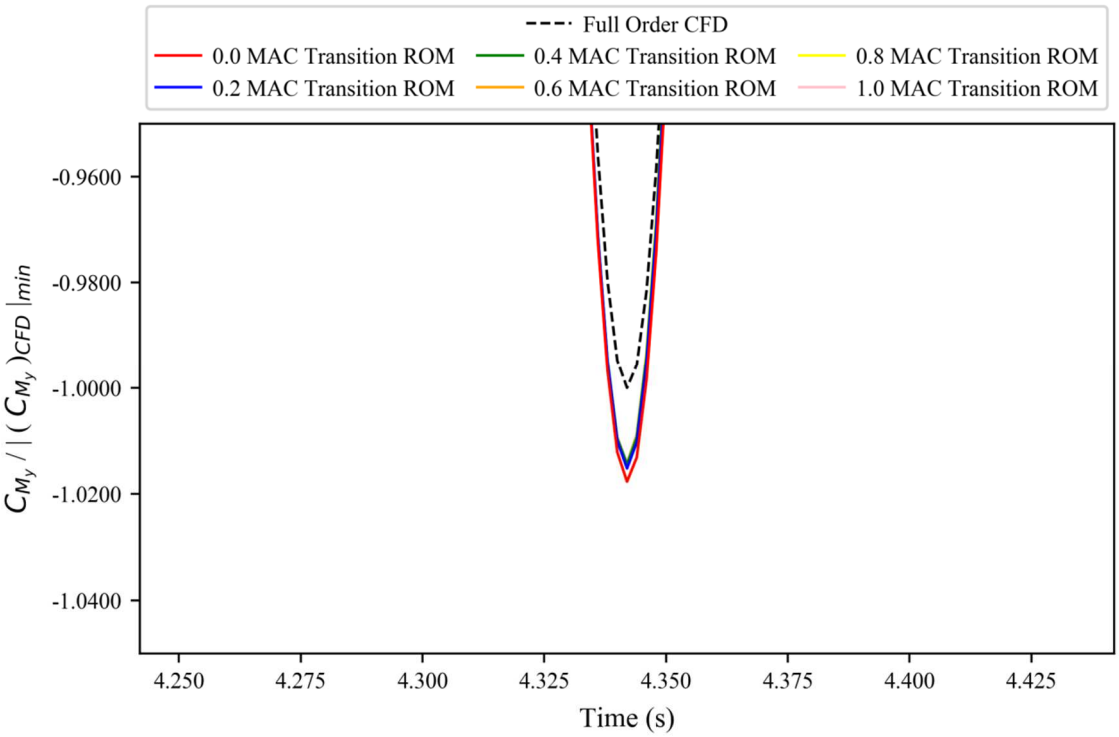

Figure 5.13. Peak response for the variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 1.

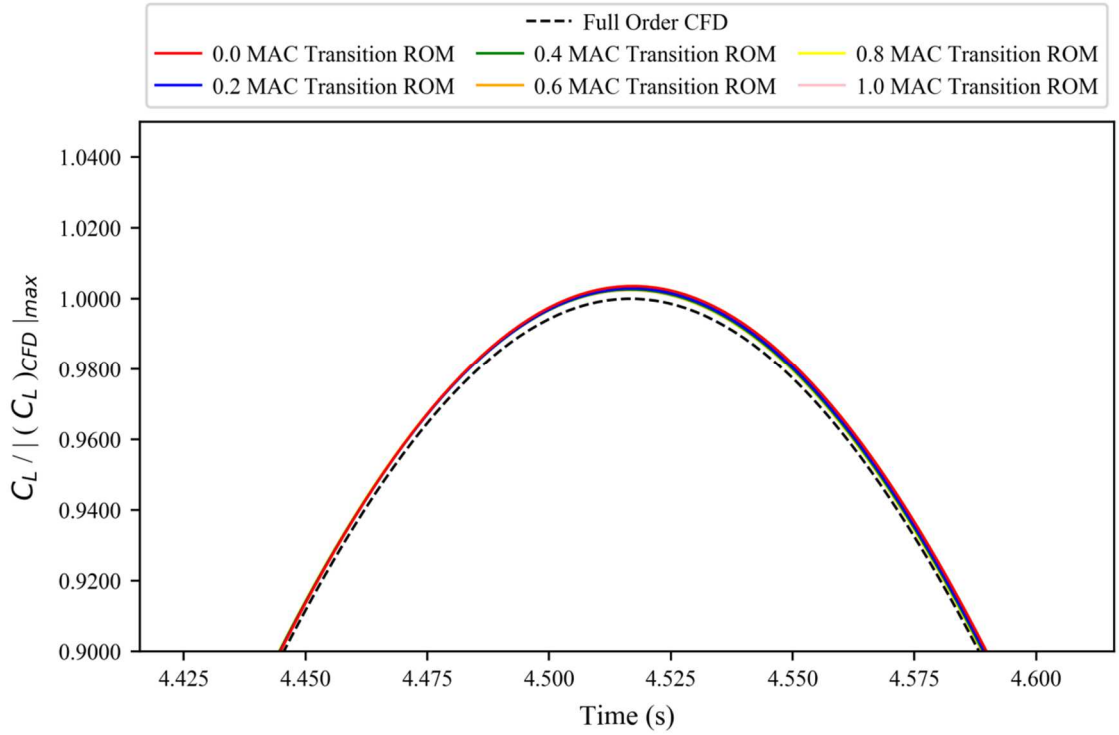

Figure 5.14. Peak response for the variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 3.

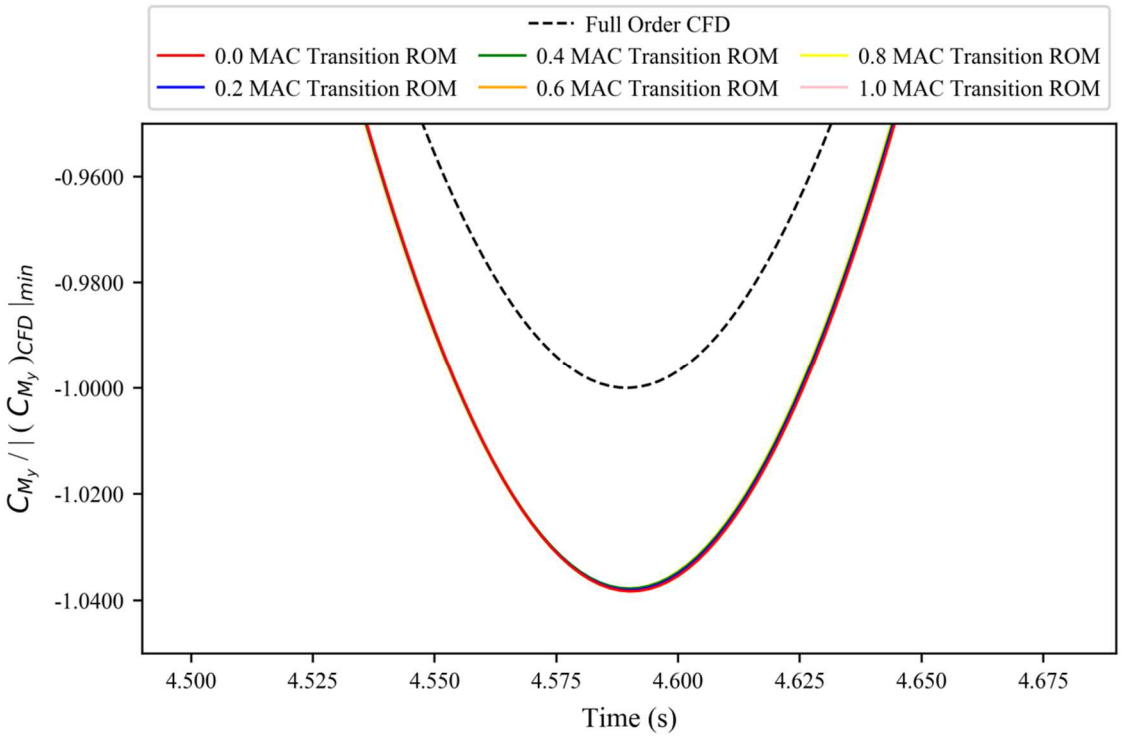

Figure 5.15. Peak response for the variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 3.

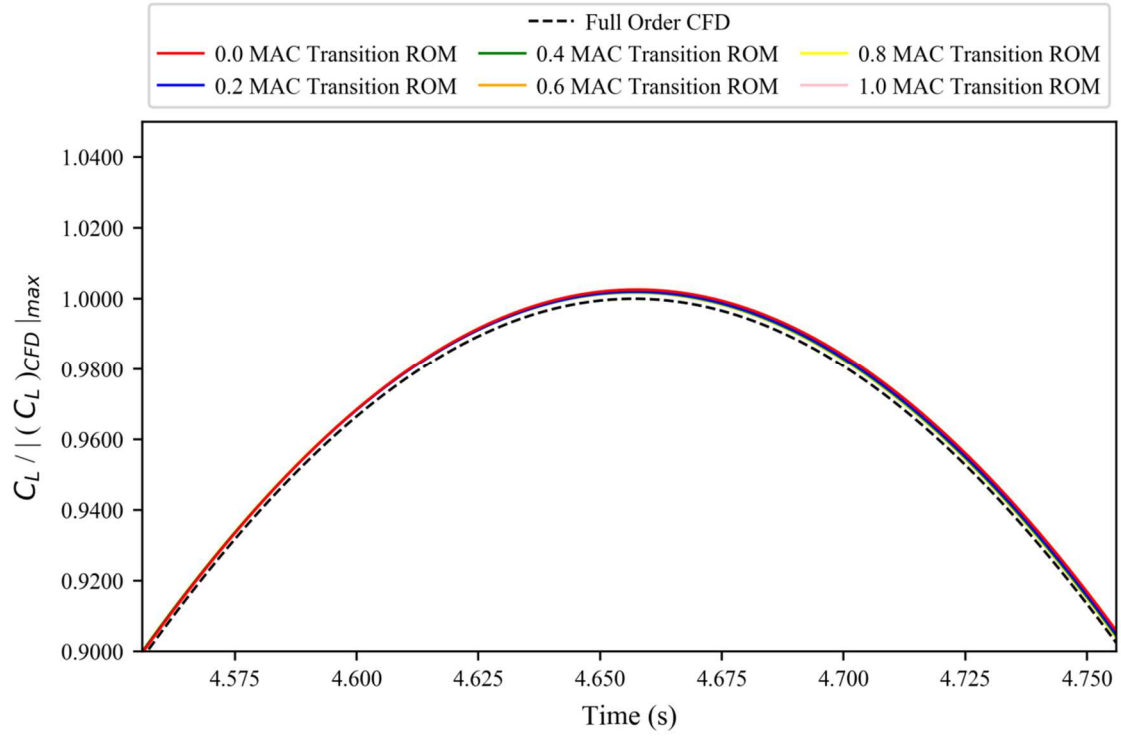

Figure 5.16. Peak response for the variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 4.

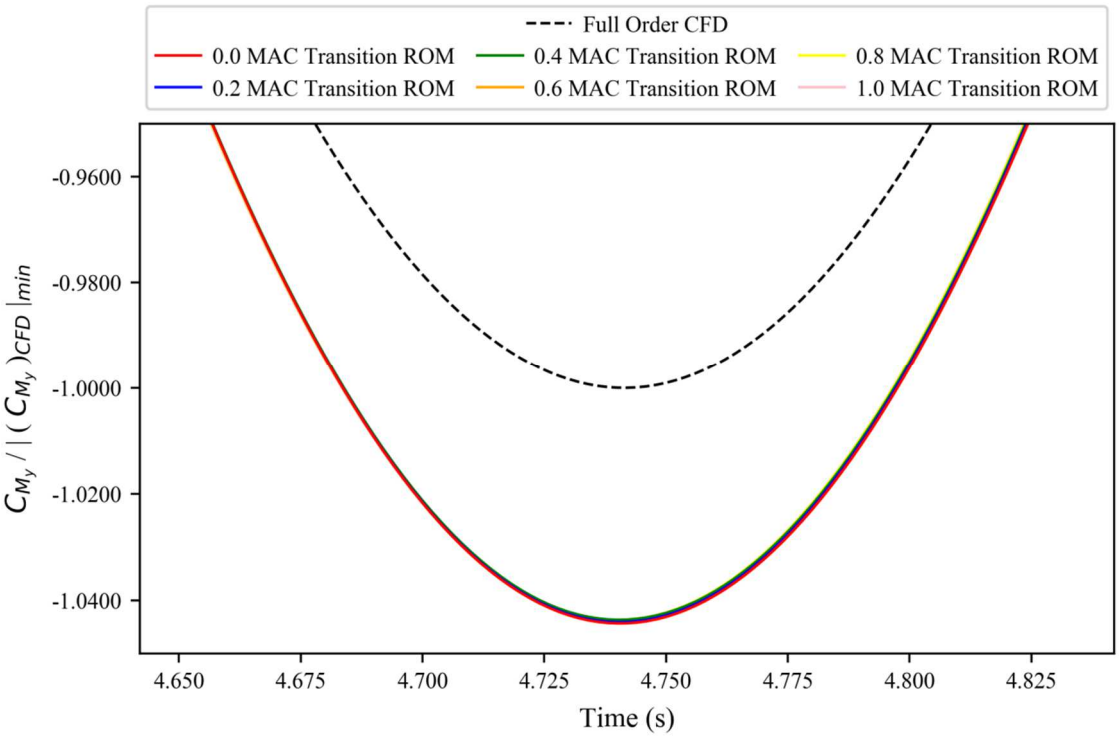

Figure 5.17. Peak response for the variable time-step based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 4.

5.2. Length of Simulations

Having verified that a time step transition size location at 0.2 MAC lengths ahead of the model remains stable, the impact that the length of sharp-edged gust has on ROM accuracy for this case is then explored. In Section 4.3.2 it was found that 2.33 model lengths (post-impact) balanced computational cost with accuracy, for the wing model. For the aircraft model, it is expected that a similar amount of data would be required, as both models are rigidly clamped on one side (i.e. there are no rigid body degrees of freedom) and so the systems are likely to settle in similar ways.

As for the previous test case, the data from a single sharp-edged gust, which transitions from large to small time steps 0.2 MAC lengths ahead of the model, was artificially cut when the gust reached different locations downstream (measured in model lengths post-impact). These sharp-edged gusts of various lengths were then used to construct the ROMs in the same manner as before; again with a ROM-size of 20.

5.2.1. Results

Testing the new ROMs on the gust cases of flight point 2, Figures 5.18-5.23 show that the ROMs built with 1.33 and 1.66 model length sharp-edged gusts produce poor results and, as expected, the ROM built with the 2.33 model length gust produces very strong results. However, slightly unexpectedly, the 2.00 model length ROM also appears to produce very accurate results. Therefore, looking at the peak response is necessary to see if there are any subtle differences between the two.

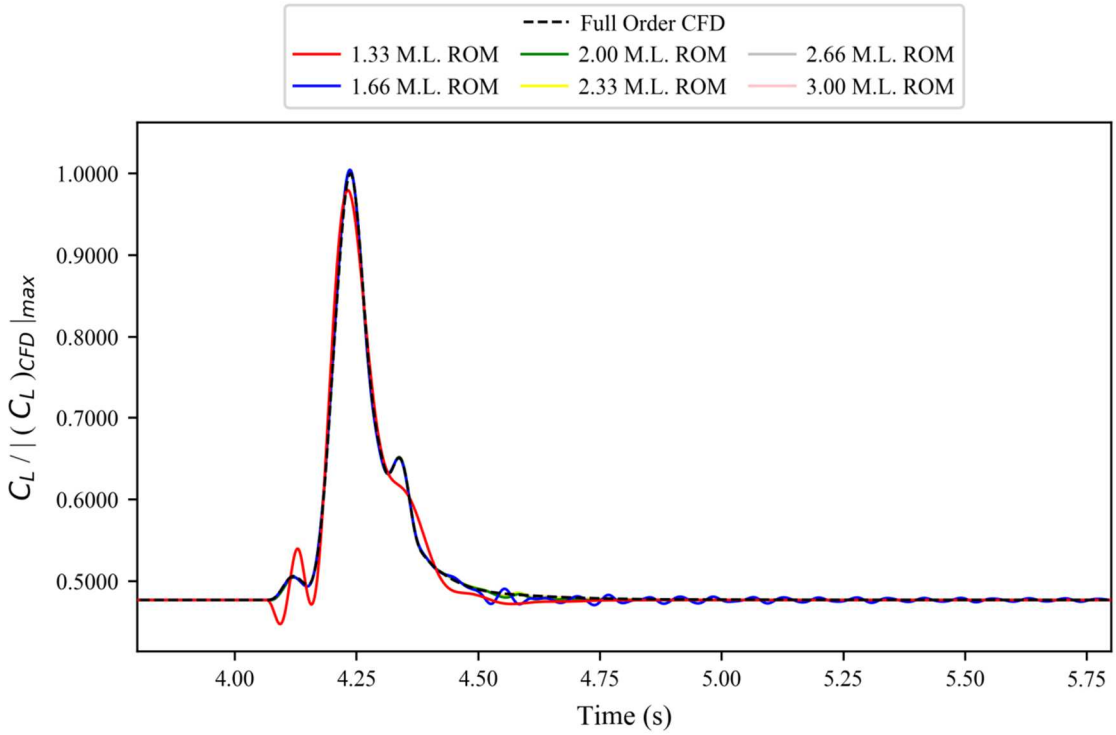

Figure 5.18. Various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 1.

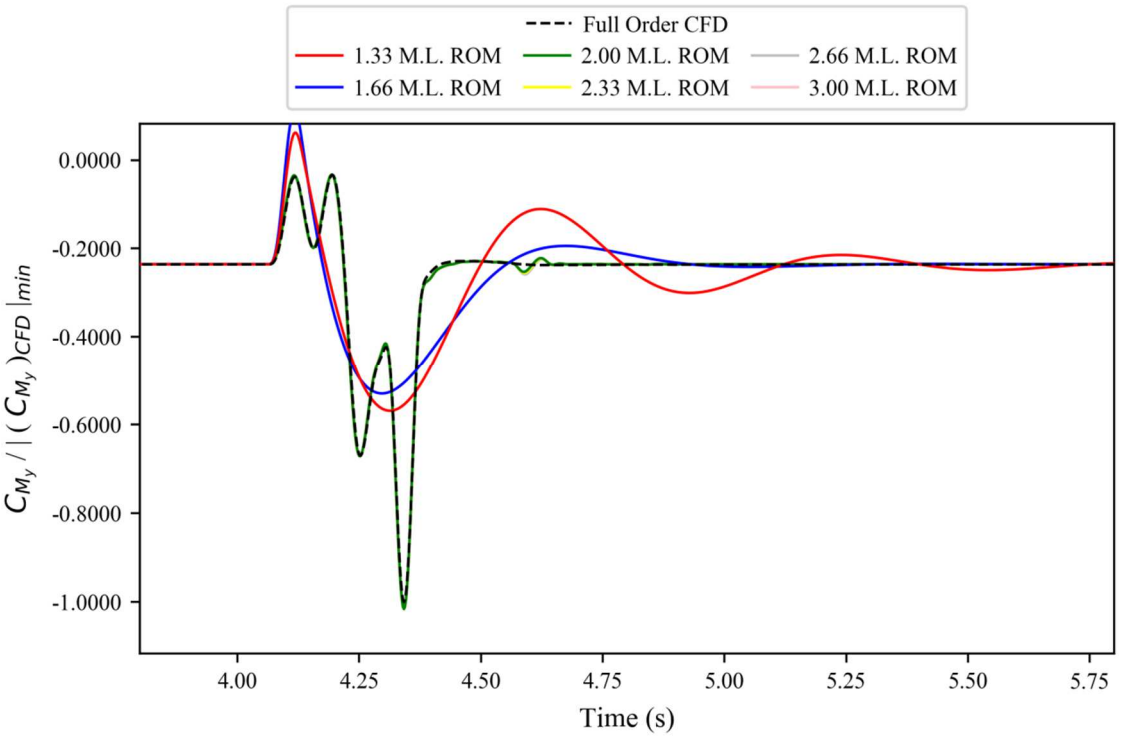

Figure 5.19. Various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 1.

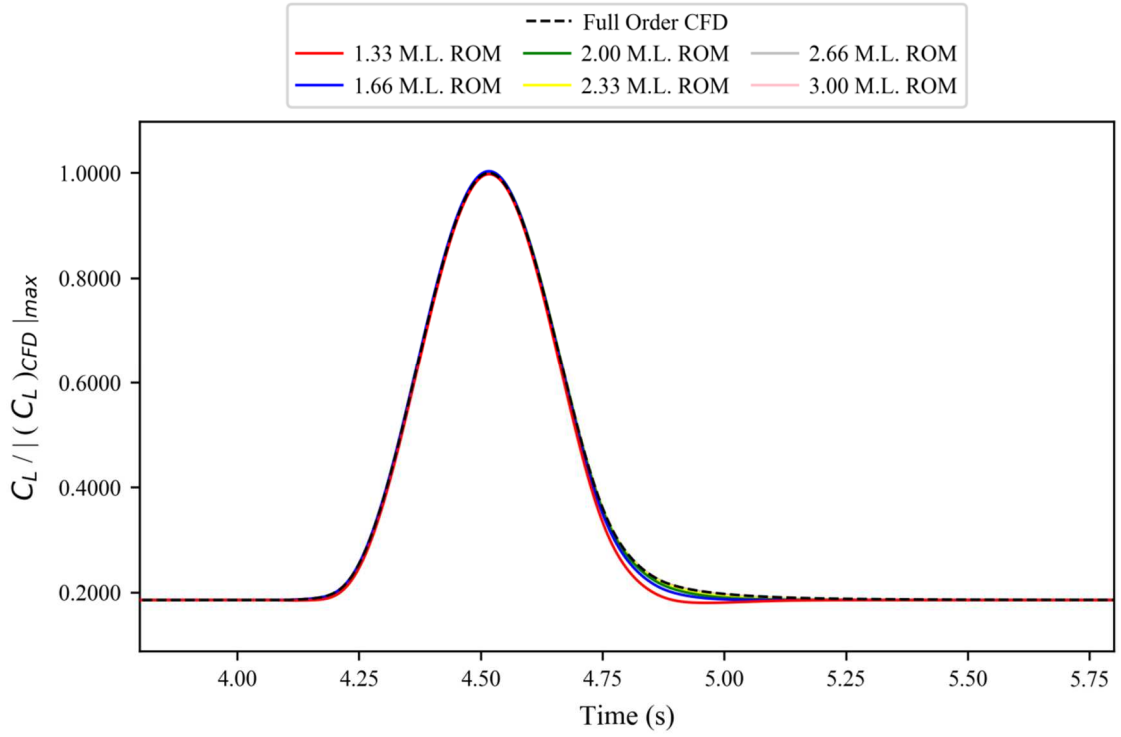

Figure 5.20. Various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 3.

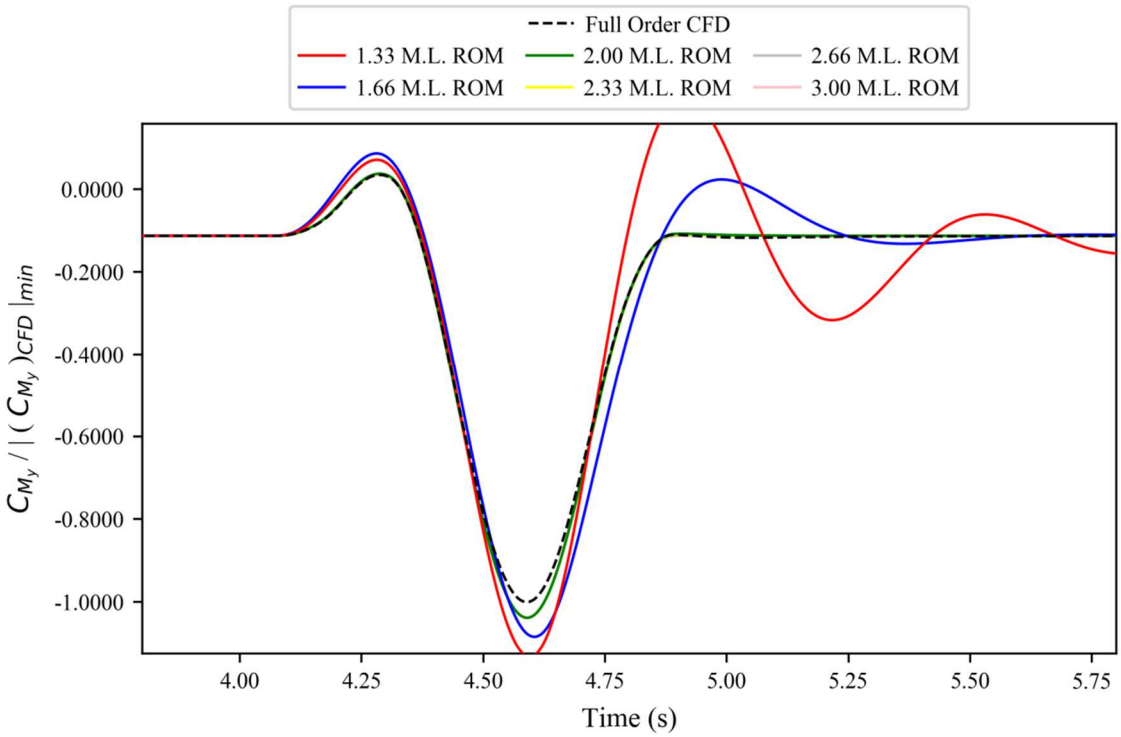

Figure 5.21. Various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 3.

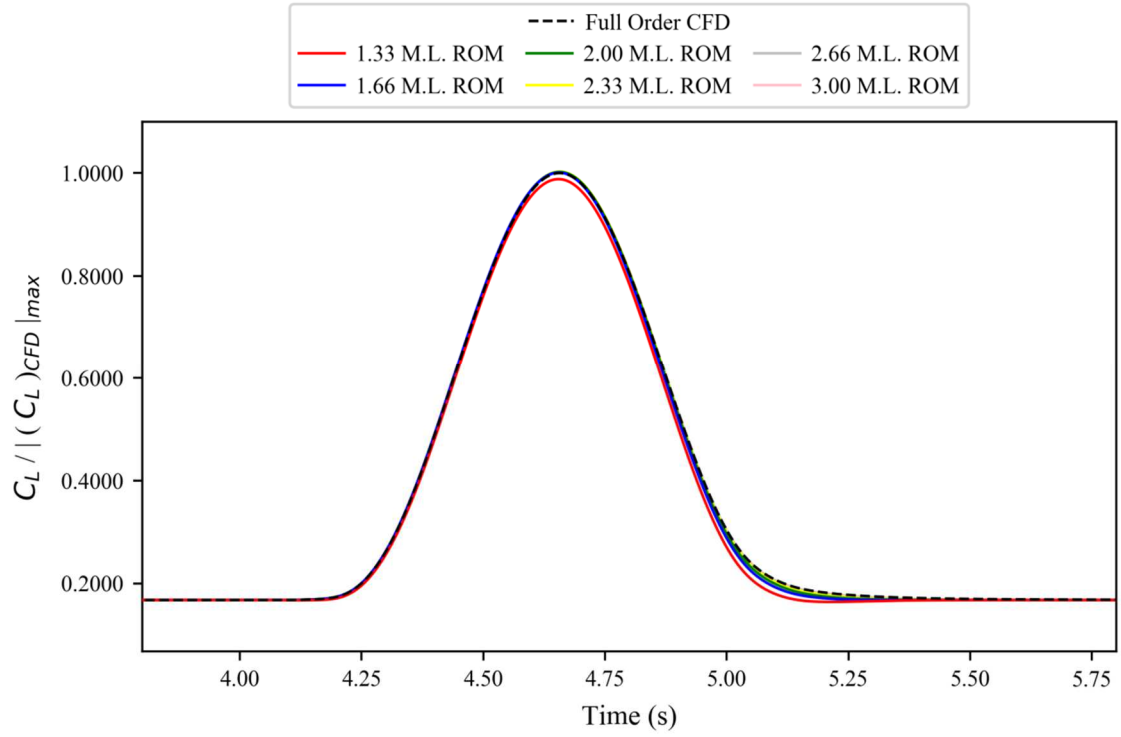

Figure 5.22. Various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 4.

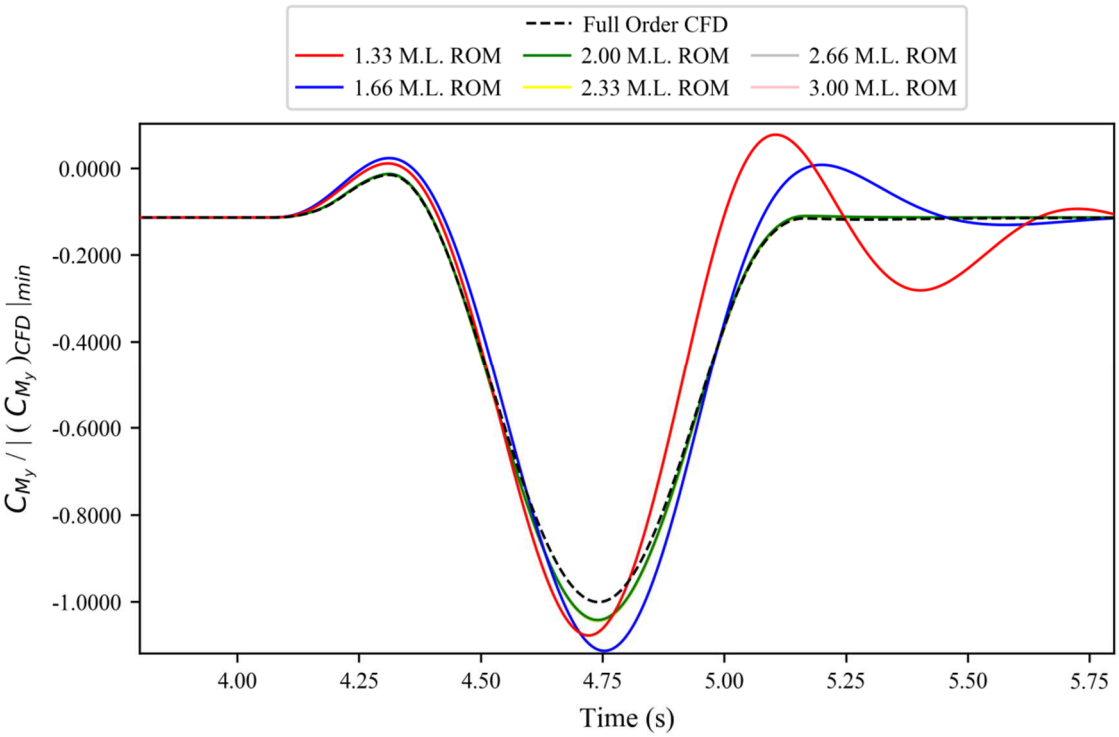

Figure 5.23. Various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 4.

Figures 5.24-5.29 mostly confirm what was already noted from Figures 5.18-5.23; there is very little difference between the ROMs built with 2.00 and 2.33 model lengths worth of data. In the worst case (the coefficient of pitching moment for gust case 4; see Figure 5.29) there is a very small difference between the two sets of results, but this is only fractionally larger than the difference between the 2.33 model length ROM and that of the 2.66 and 3.00 ROMs.

It would be entirely reasonable, based on the results shown, to go with 2.00 model length sharp-edged gusts. However, as before gust robustness should be taken into account. As 2.33 model lengths would appear to ensure the ROM produces accurate results for models with no rigid body degrees of freedom, it is perhaps the best option to take forward; balancing ROM robustness with computational efficiency.

It should be highlighted that some of the ROM results are almost impossible to distinguish from each other; even in these peak responses. This is perhaps most notable in Figure 5.24, where the data for 2+ model lengths of data effectively hidden behind the data for 1.66 model lengths.

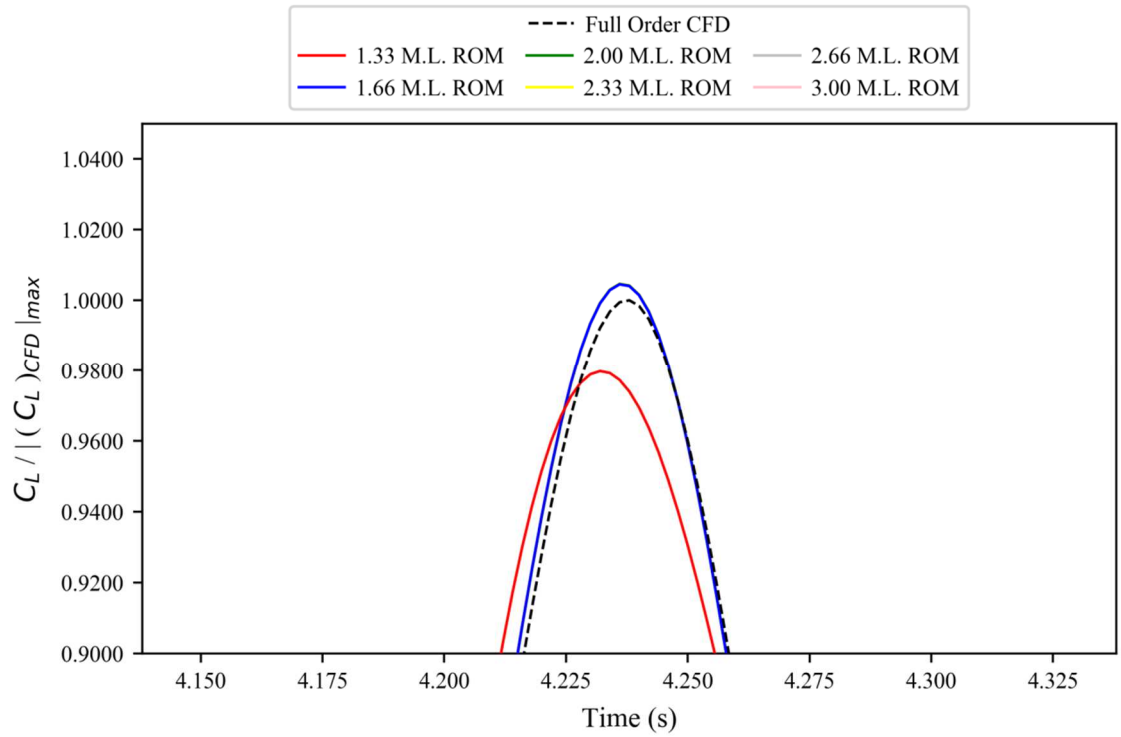

Figure 5.24. Peak response for the various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 1.

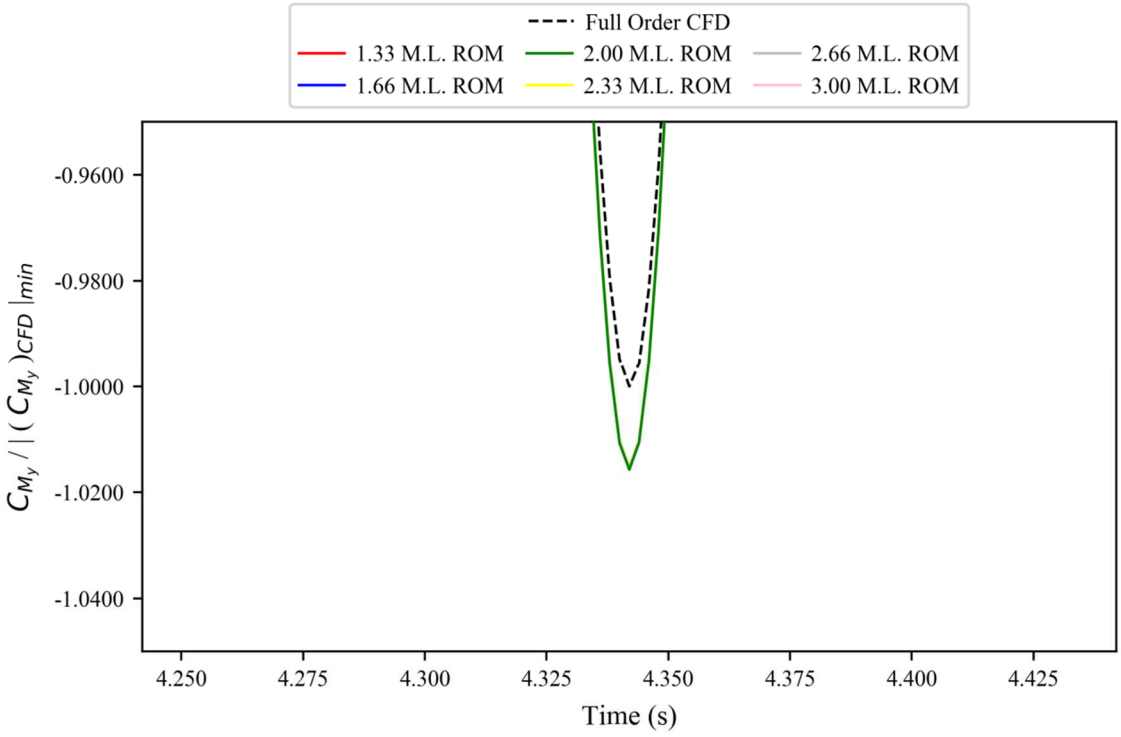

Figure 5.25. Peak response for the various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 1.

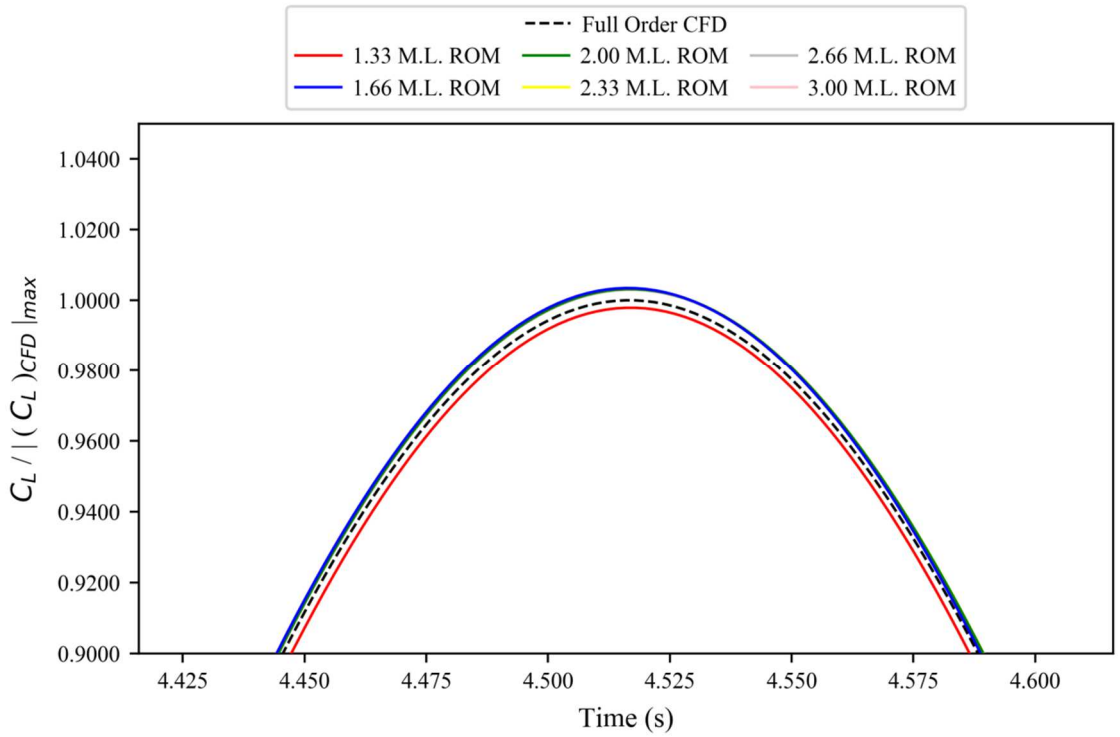

Figure 5.26. Peak response for the various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 3.

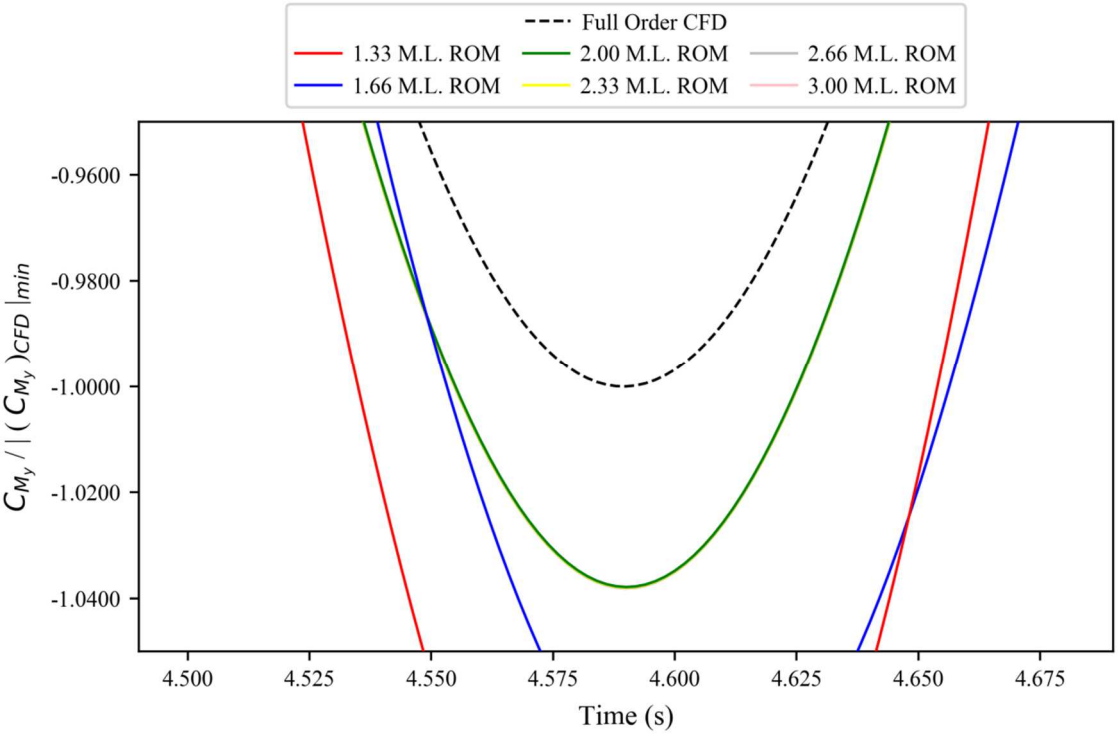

Figure 5.27. Peak response for the various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 3.

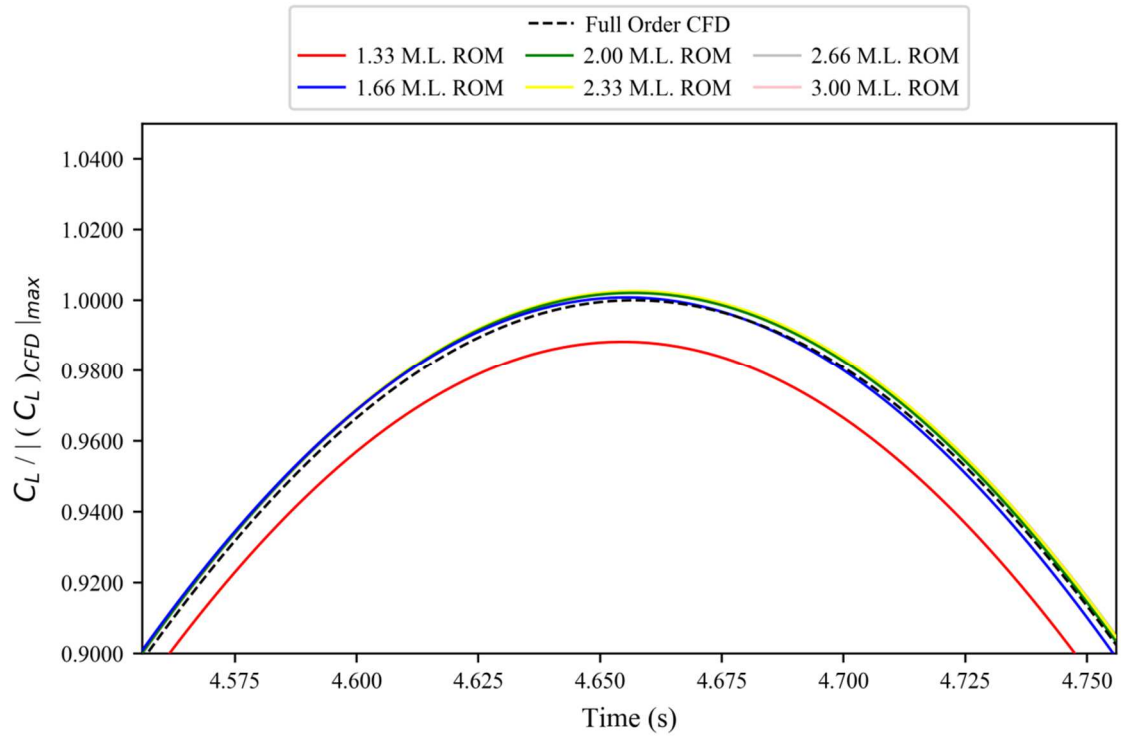

Figure 5.28. Peak response for the various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 2, gust case 4.

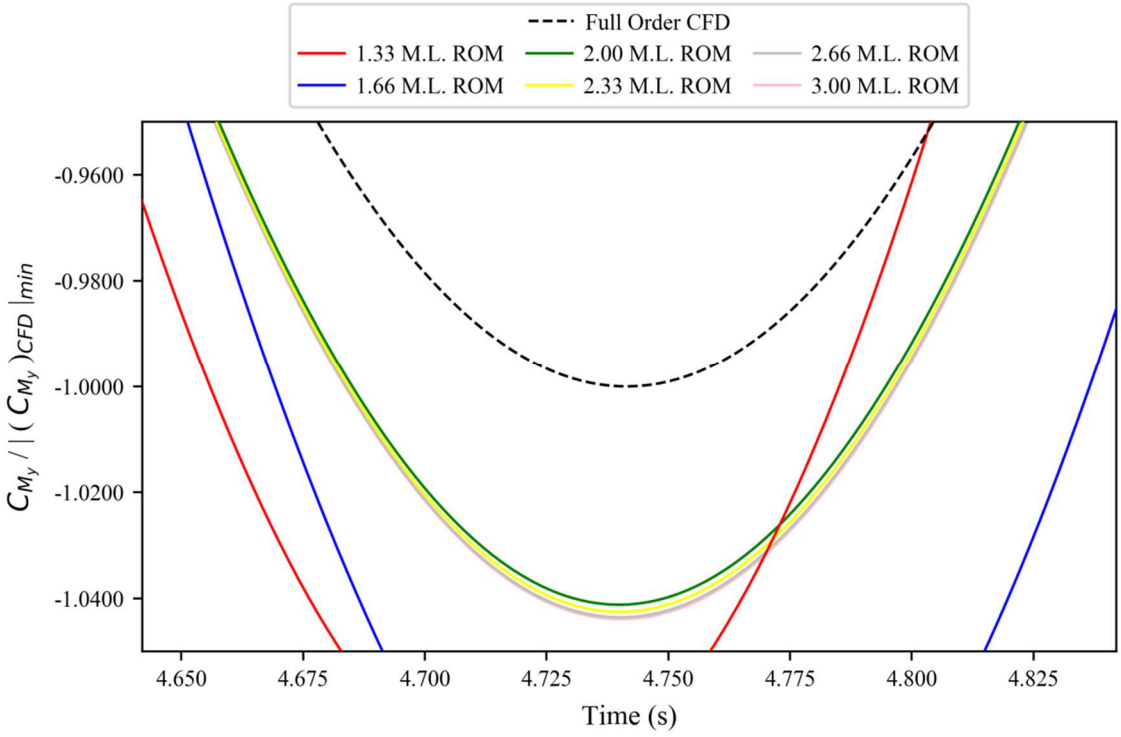

Figure 5.29. Peak response for the various sharp-edged gust length based ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 2, gust case 4.

5.3. Restarting Problem

The current ROM method, like all those used to produce results from Section 4.5 onward for the inviscid wing case, used restarting; which ensures a stable discrete ROM is found. Normally this is a rather straight forward process; however a problem was detected during some peripheral tests. These tests were generic tests that were occasionally carried out to check for any unexpected behaviour when setup parameters such as ROM size, gust type, etc. were changed. This is normally done by simply trying to recreate the system's input response (so for these cases, the step-down response) with varying setup parameters.

In this particular test, a wide range of ROM sizes were being looked at, ranging from the very small (~2) to the very large (~500, using a longer sharp-edged gust than the one taken forward in Section 5.2.1). During these tests, it was found that restarting would occasionally produce a ROM that was technically stable (i.e. all eigenvalues resided within the stable region), but were still often highly inaccurate having lost the link to the original physics.

5.3.1. No Stabilising Method Applied

To get a clear understanding of how the restarting method can fail, it is first necessary to look at a specific example ROM using no stabilisation method (such as restarting) applied. Here, a large ROM size of 220 was created using 453 Markov parameters.

Figure 5.30 shows a relatively normal set of initial eigenvalues for the discrete ROM of a system; with the majority of the eigenvalues falling within the discrete stable region, but a handful falling marginally outside. At this point there is nothing particularly untoward, and nothing that would suggest restarting would have a problem stabilising this particular ROM.

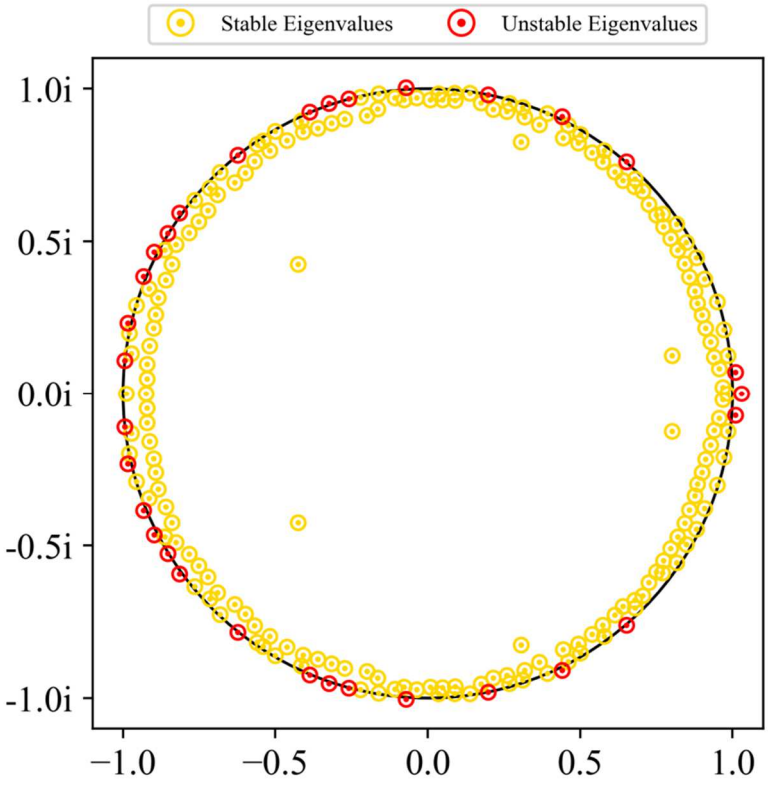

Figure 5.30. Eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with no stability method applied.

Figure 5.31 shows the recreation of the input step-down response using this unstable ROM. For the most part, the ROM is relatively good, matching the input for its entire length. However if continued past this point, the ROM's instability starts to take hold and the response undergoes a sudden exponential growth.

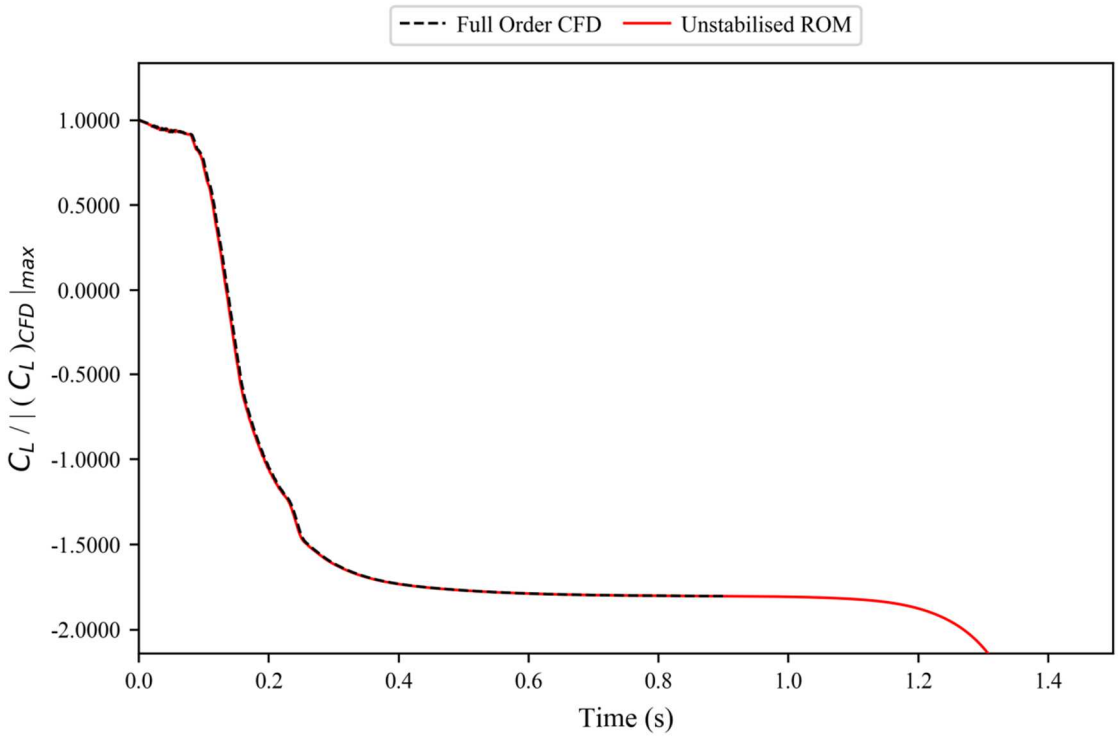

Figure 5.31. Recreation of input step-down response for the coefficient of lift of the generic wide-bodied aircraft model; with no stability method applied.

5.3.2. Restarting Method Applied

Normally, for a case like this removing the unstable eigenvalues would be straightforward via restarting; the method previously incorporated into the ROM (see Section 4.5) would automatically take over on the detection of one or more unstable eigenvalues, and would go on to produce new, stable system matrices. However, in this particular case this does not happen. The process led to 42 restarts of the ROM (an unusually high number); which caused the ROM size to be reduced from the requested 220 to 187 (it should be noted here that increasing the number of Markov parameters available actually resulted in additional restarts and a smaller final ROM size).

The starting eigenvalues were the same as for the ROM with no stability method applied (see Figure 5.30). After restarting is applied, it can be seen in Figure 5.32 that the final set of eigenvalues are quite unusual. Typically, the number of eigenvalues near the origin would be very small (a few if any), however in this case, there is a large amount of eigenvalues clustered around this point. To understand what has occurred, it is worth looking at the eigenvalues from the previous 15 restarts (restarts 27 to 41).

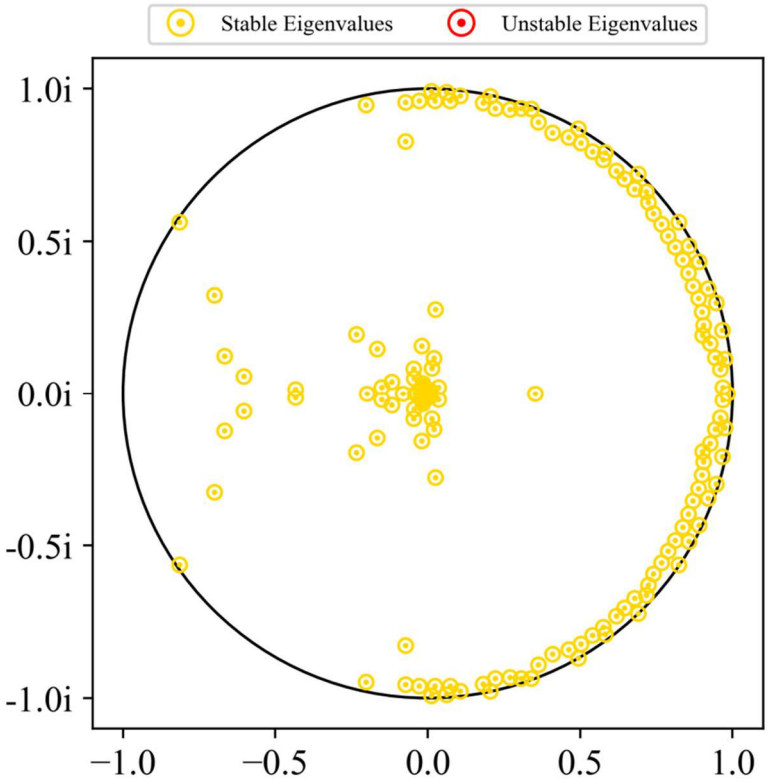

Figure 5.32. Final eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

Figures 5.33-5.47 and 5.32 show how the restarting seems to have effectively created a "blind spot", around which no additional eigenvalues are found. This is fine when it comes to the unstable values; however the stable eigenvalues are also forced to be found increasingly further away from this area. The blind spot appears to begin in restart 29 (see Figure 5.35) with a small area in the region of -1.0 ± 0.1i. This region then appears to grow (predominantly along the edge of the stable region) through each subsequent restart, until the final one is stable but with next to no eigenvalues found near the edge of the stable region from 0 to -1 on the real axis.

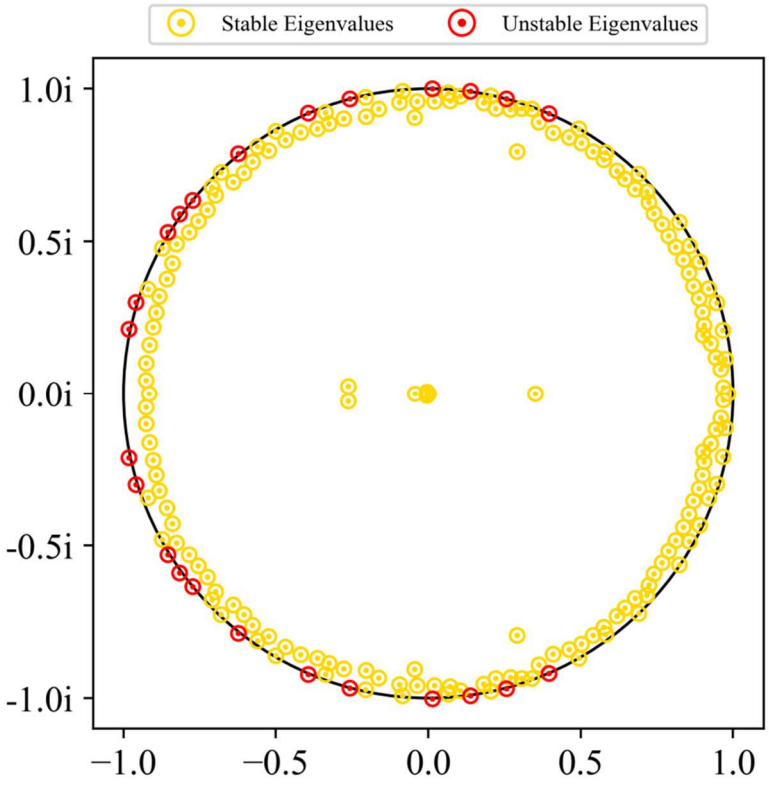

Figure 5.33. 27th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

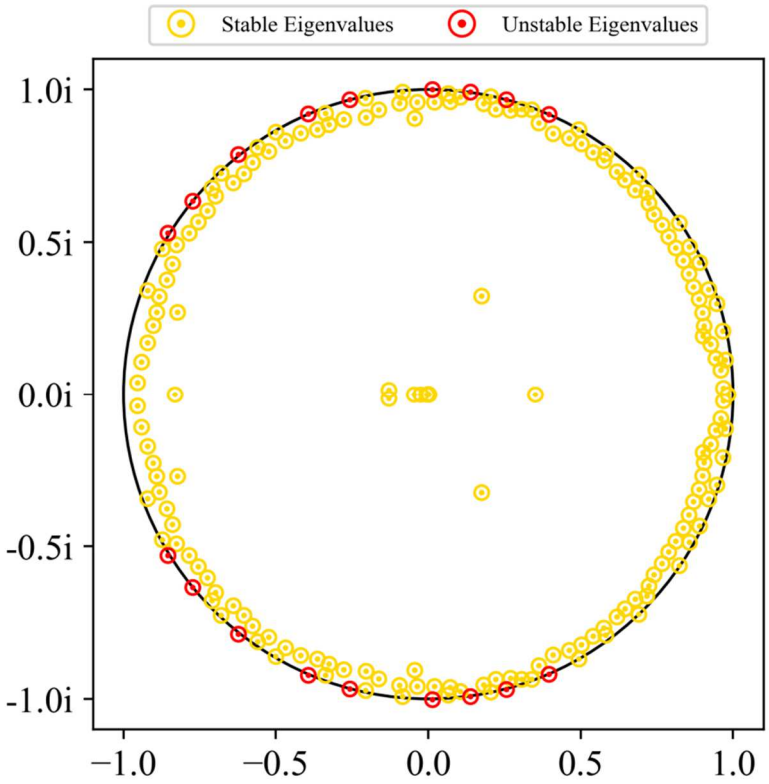

Figure 5.34. 28th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

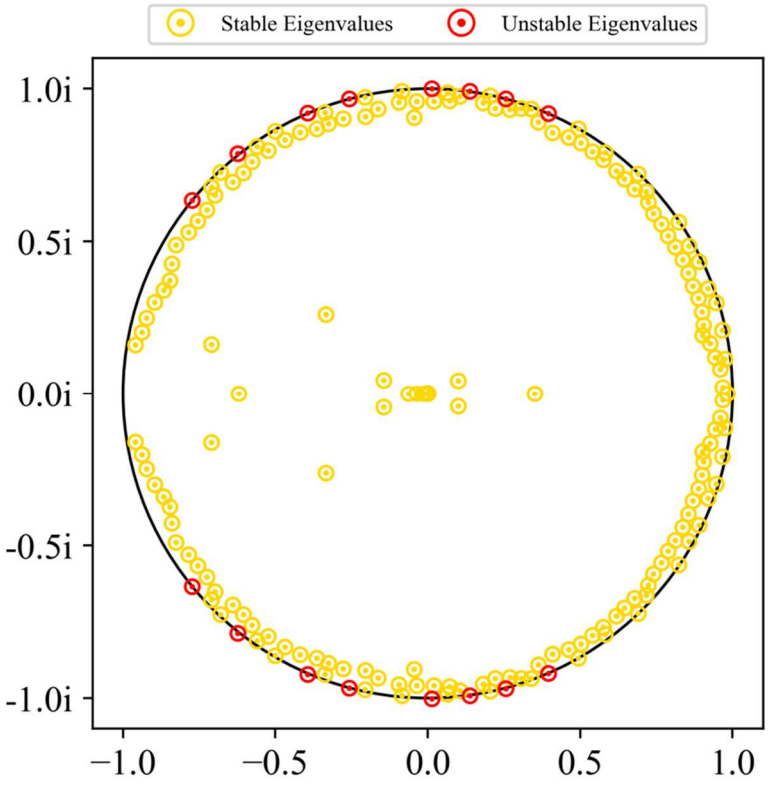

Figure 5.35. 29th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

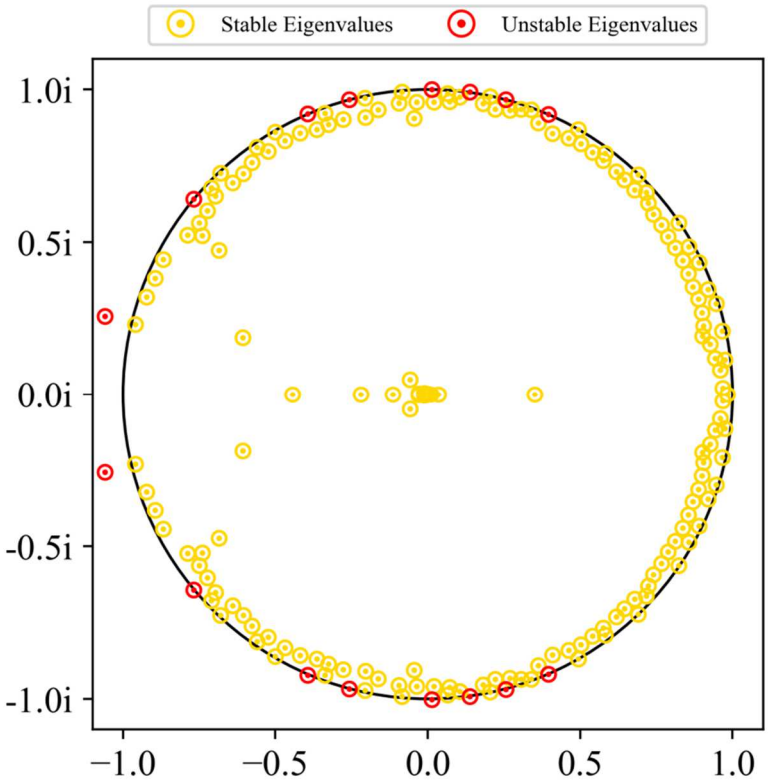

Figure 5.36. 30th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

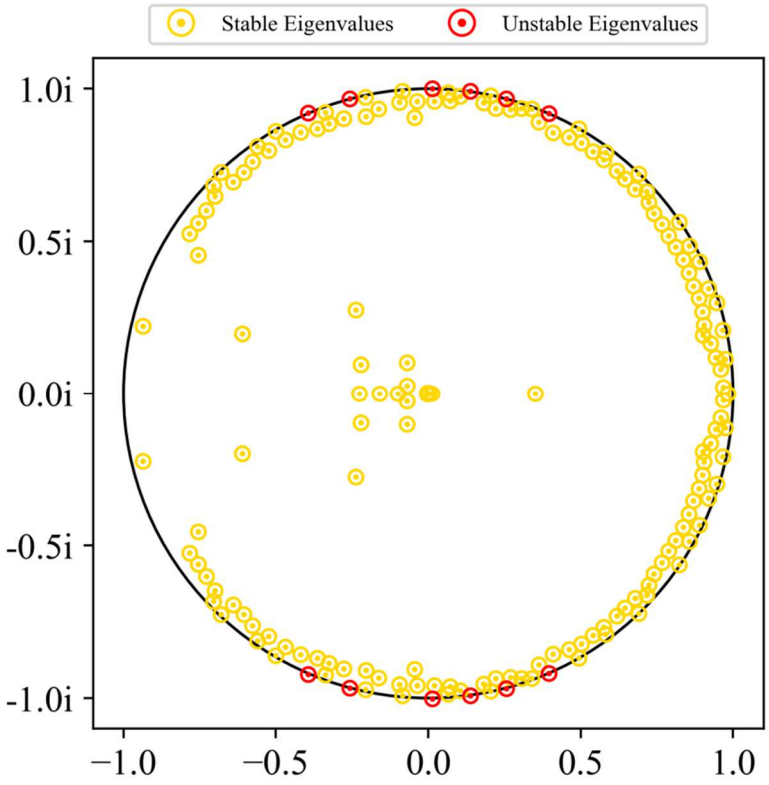

Figure 5.37. 31st set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

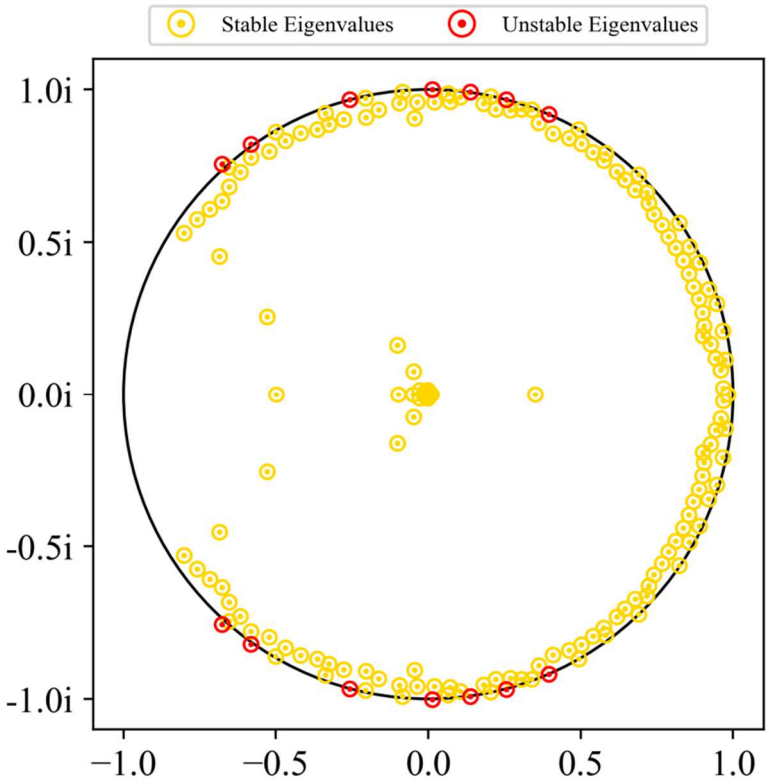

Figure 5.38. 32nd set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

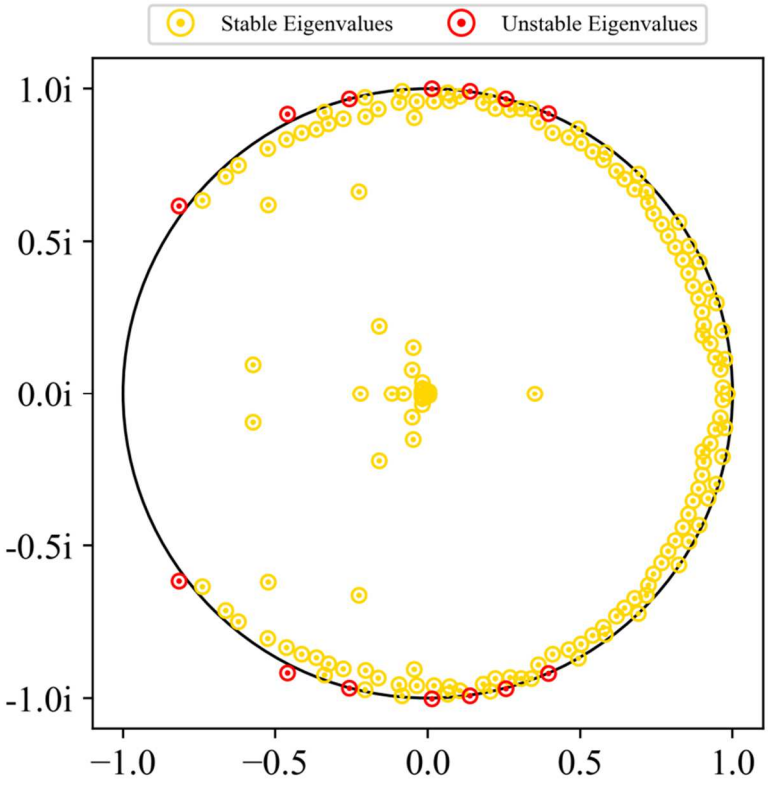

Figure 5.39. 33rd set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

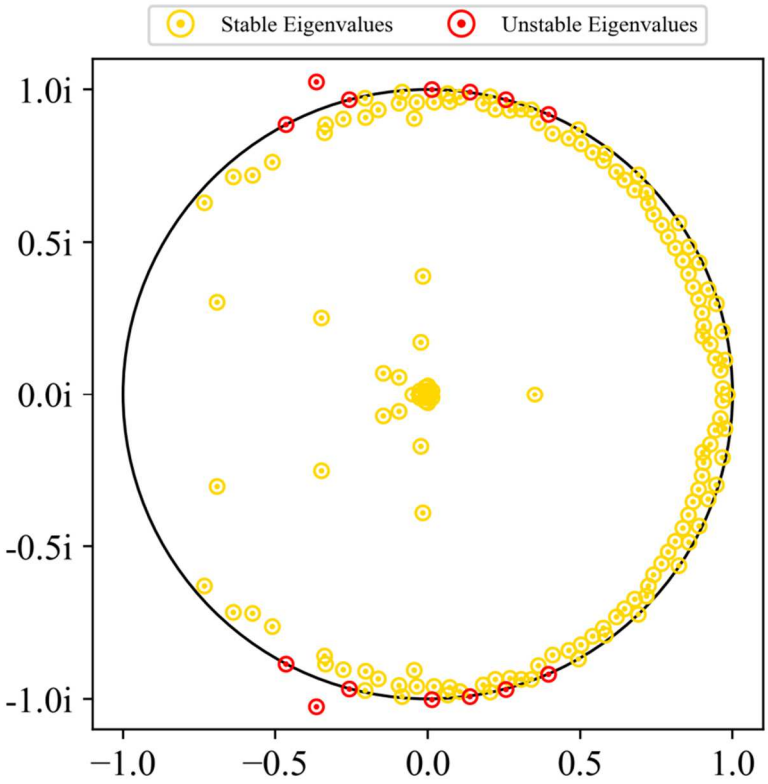

Figure 5.40. 34th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

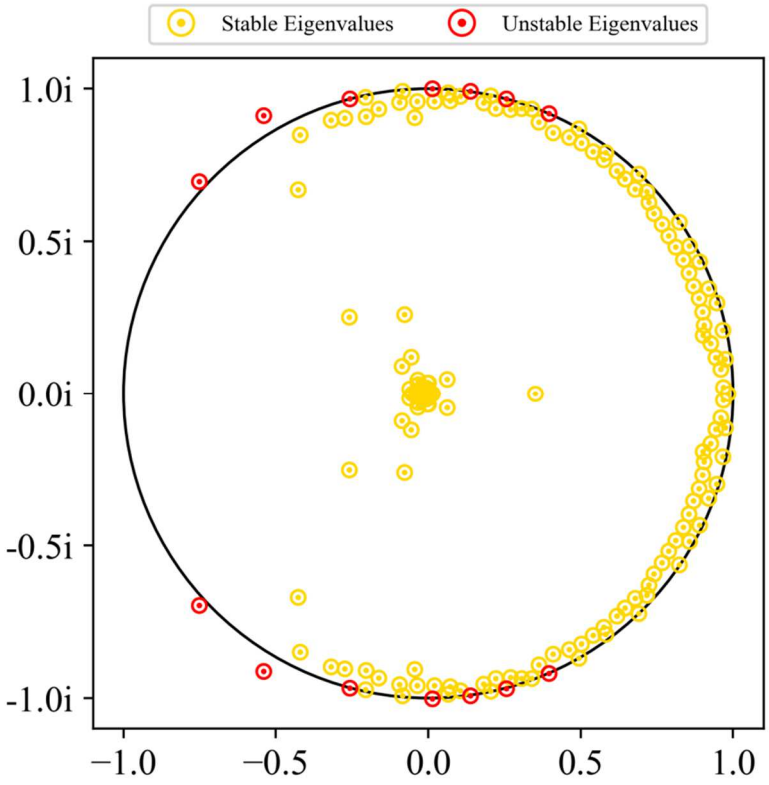

Figure 5.41. 35th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

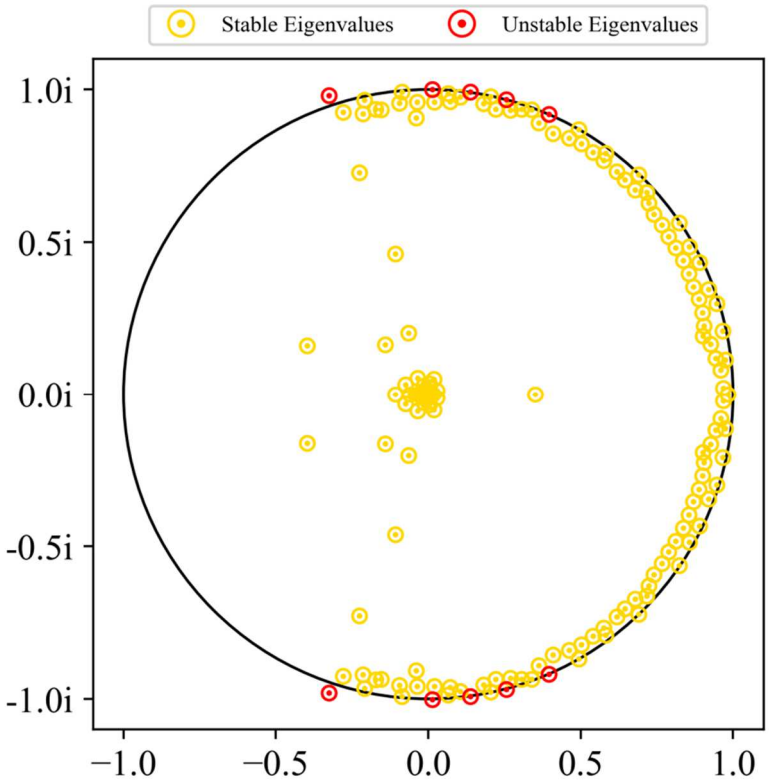

Figure 5.42. 36th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

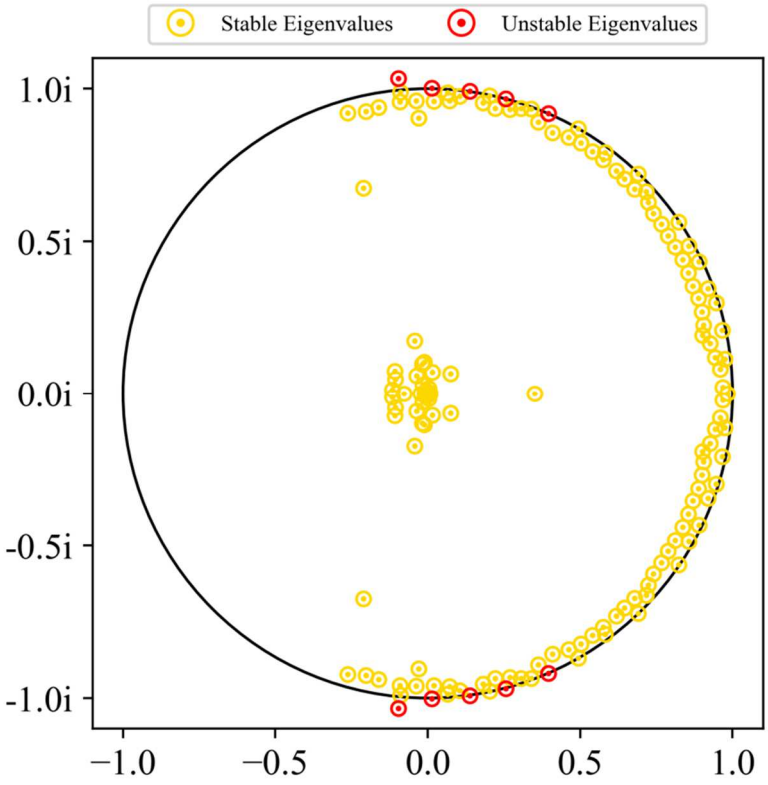

Figure 5.43. 37th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

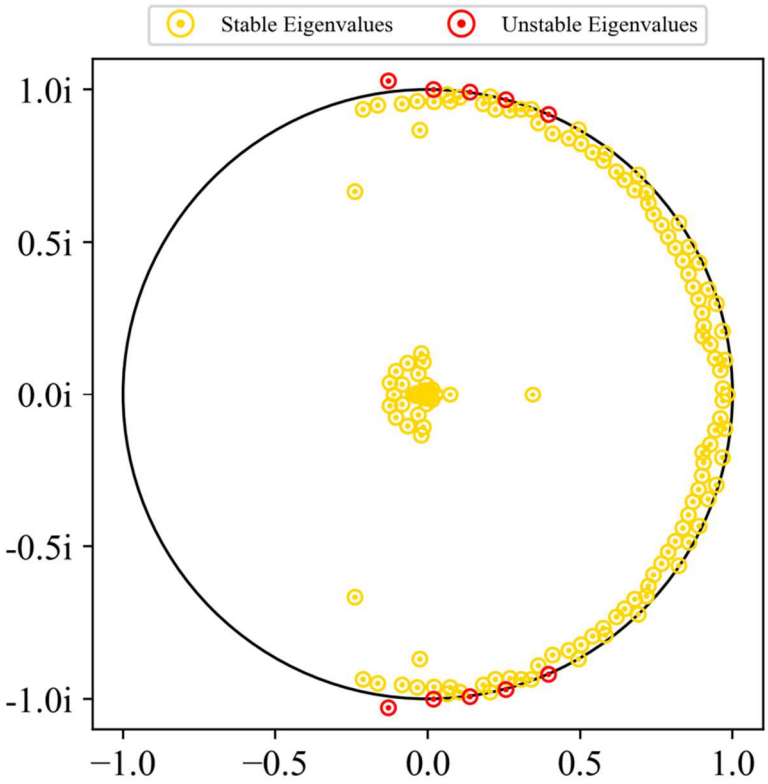

Figure 5.44. 38th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

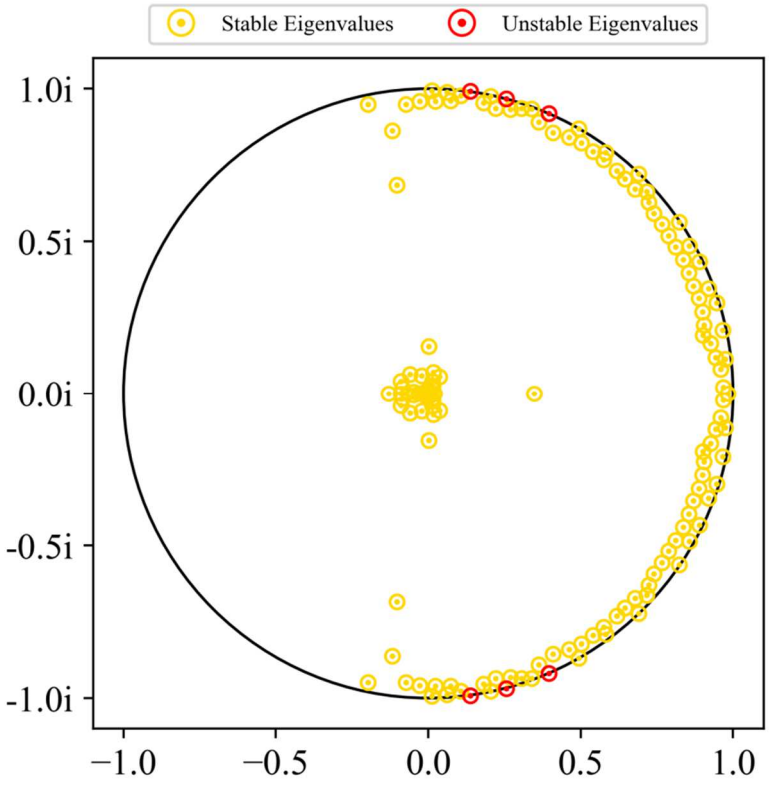

Figure 5.45. 39th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

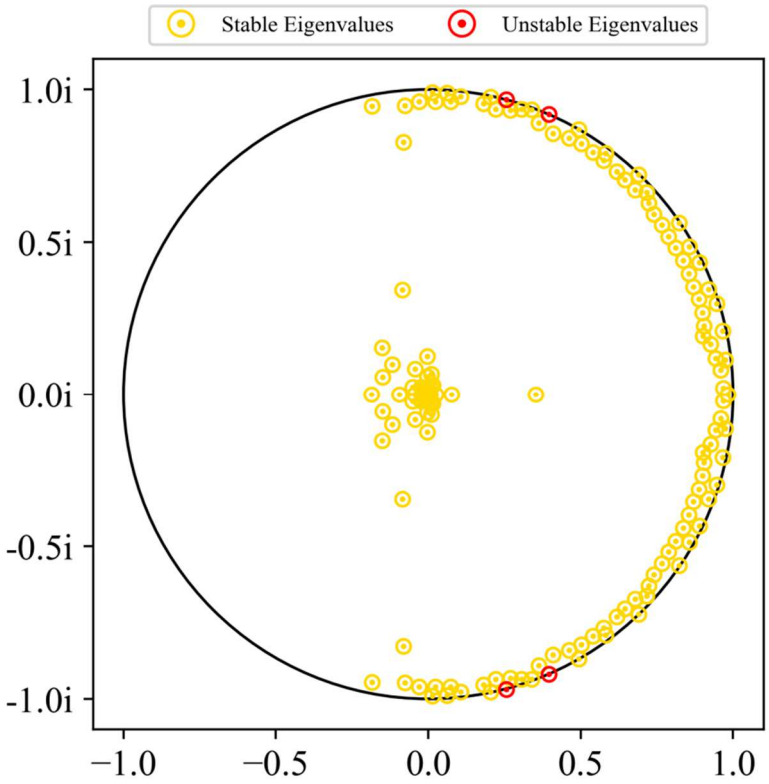

Figure 5.46. 40th set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

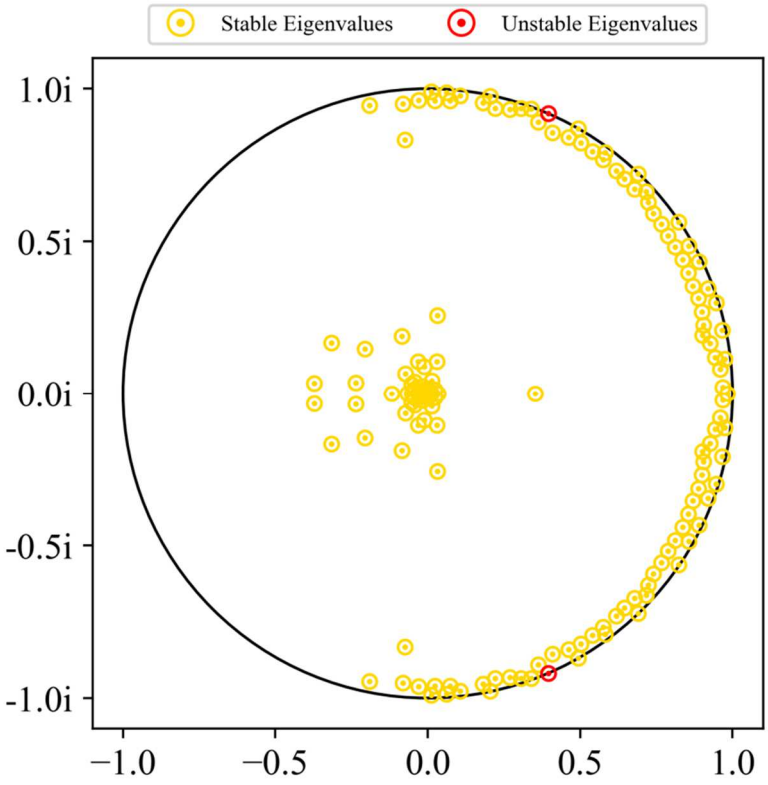

Figure 5.47. 41st set of eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the restarting stability method applied.

Figure 5.48 shows the simulation of the input step-down response using this restarted ROM. Despite being technically stable, it is immediately apparent that the response shows all the hallmarks of an unstable response; notably, oscillation and exponential growth. The likely explanation is that the restarting method, having seemingly created this blind spot for eigenvalues, has lost critical information about the system response; which in turn leads to a meaningless output which lacks many of the dominant behaviours of the system.

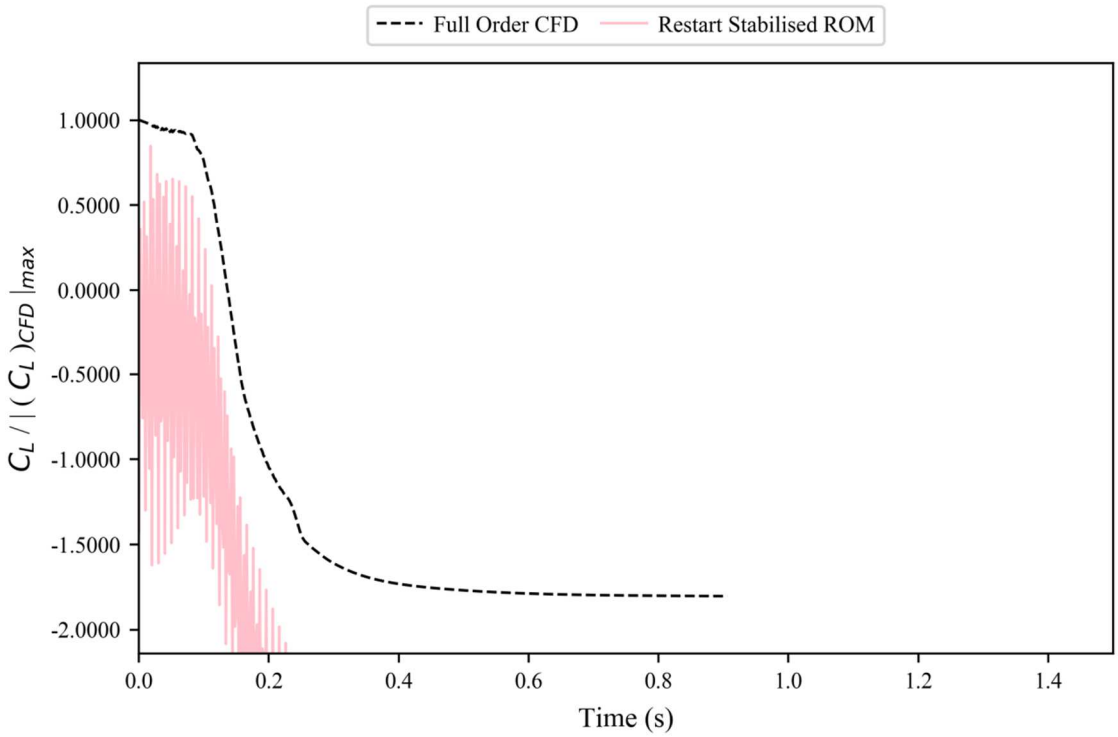

Figure 5.48. Recreation of input step-down response for the coefficient of lift of the generic wide-bodied aircraft model; with the restart stability method applied.

5.3.3. Schur Mirroring Stability Method Applied

An alternative stability method to restarting is the Schur mirroring technique. As noted in Section 3.4.2, this method was described by McKelvey *et al* [98] and uses Schur decomposition to obtain the system eigenvalues, before applying a simple mathematical manipulation (see Eqn. (3.48)) to effectively reflect the unstable eigenvalues back within the stable region. These new, stable, eigenvalues are then used to reconstruct a new set of system matrices.

One subtle, yet notable, difference between the restarting and Schur mirroring methods is the number of iterations required for each. Restarting is only applied to one unstable eigenvalue at a time, and can therefore often require multiple iterations to obtain a completely stable set of eigenvalues. However, the Schur mirroring method is applied to all unstable eigenvalues at once and so only needs to be carried out a single time. As such, for Schur mirroring only a starting and end set of eigenvalues are produced.

The starting set of eigenvalues is the same as with the previous two cases and shown in Figure 5.30 (no stabilisation and restarting). Figure 5.49 however shows that, unlike the restarting method, all the final eigenvalues are not only located within the stable region,

but are also distributed relatively evenly around the edge of the stable region. This is because the unstable eigenvalues have simply been reflected back into the stable region, with the stable eigenvalues left as they were.

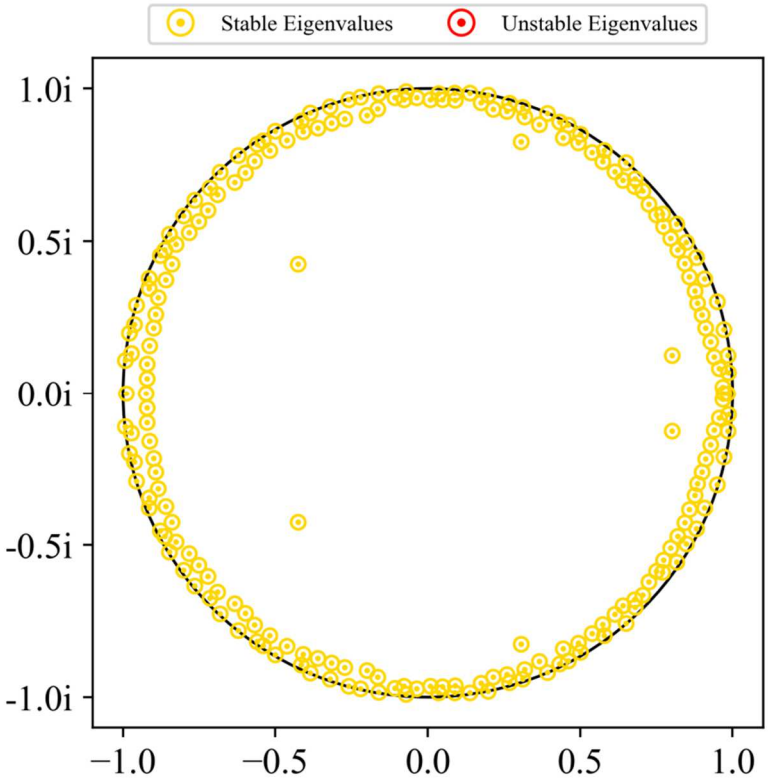

Figure 5.49. Final eigenvalues for a discrete ROM constructed to model the system's coefficient of lift; with the Schur mirroring stability method applied.

Figure 5.50 shows that this new ROM, created using Schur mirroring, can simulate the step-down response well; with none of the instability from the non-stabilised ROM present.

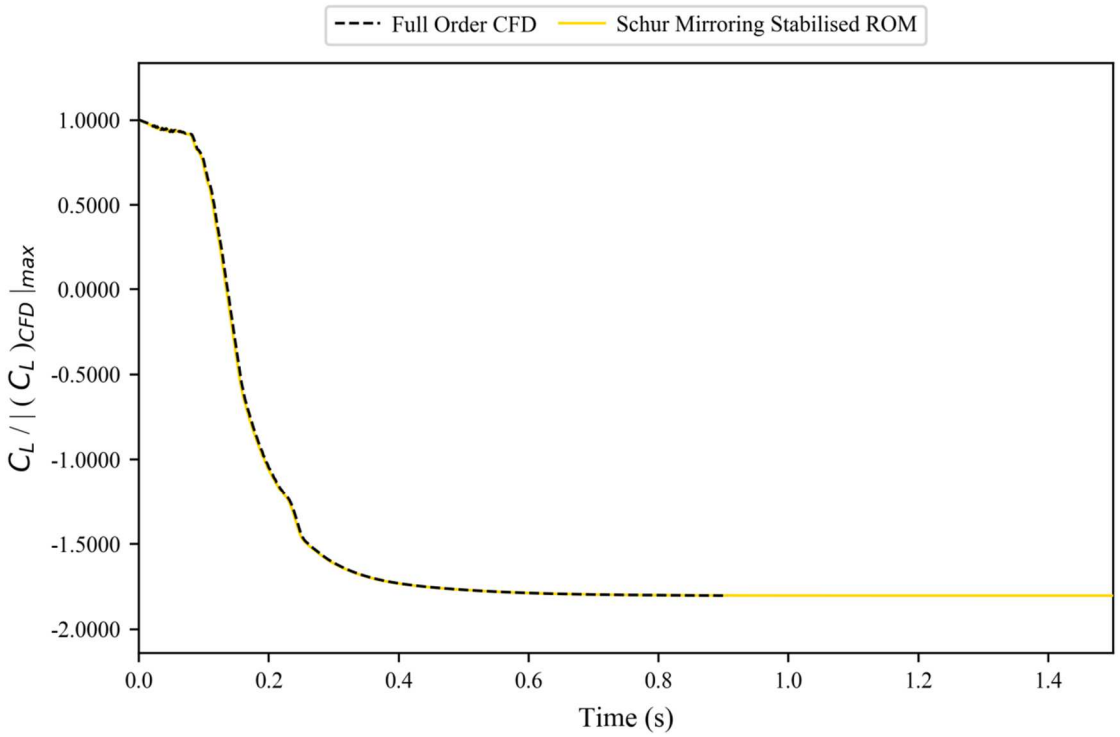

Figure 5.50. Recreation of input step-down response for the coefficient of lift of the generic wide-bodied aircraft model; with the Schur mirroring stability method applied.

The Schur mirroring method has been shown to offer a more robust alternative to restarting. However, it should also be noted that in cases where restarting does not fail, it is a preferable method. This is because restarting uses additional information to obtain a completely new set of eigenvalues. By contrast, Schur mirroring uses a mathematical manipulation of the original eigenvalues. As a result of being a non-physics based stability method, Schur mirroring can't be assumed to work on all cases; despite showing very good robustness on the cases used in this thesis. Conversely, restarting is physics based so can be assumed to work on most cases; albeit with the caveat that it has been demonstrated to occasionally fail. As restarting failures are extremely obvious it is therefore reasonable to use restarting as the default stability method, with an automatic switch to Schur mirroring if restarting is found to have failed.

5.4. Additional Results

In this chapter of the thesis it was shown that the ROM taken forward from the end of the Chapter 0 was also valid for the viscous aircraft model with no rigid body degrees of freedom. The ROM balances robustness with computational efficiency by simulating a single sharp-edged gust of magnitude $1ms^{-1}$, which transitions from large time steps to

small ones 0.2 MAC lengths ahead of the leading edge and continues to run for 2.33 model lengths post-impact. This sharp-edged gust is then used to create a step-down ROM; with a ROM-size of 20 and stability is ensured via restarting and/or Schur mirroring.

Whilst this ROM has been extensively demonstrated on the four gusts of flight point 2 (see Figures 5.18-5.23) it is worth exploring the other four gust points laid out in Table 5 in order to gain a more complete understanding on how well the ROM performs on the whole aircraft model. The results for flight points 1 (Figures 5.51-5.58), 3 (Figures 5.59-5.66), 4 (Figures 5.67-5.74) and 5 (Figures 5.75-5.82) all show good matches to the full order CFD. It can be seen that as the gust length increases, the accuracy does decrease slightly (most notable in the coefficient of pitching cases), however even for the largest gust the accuracy is still good.

5.4.1. Flight Point 1

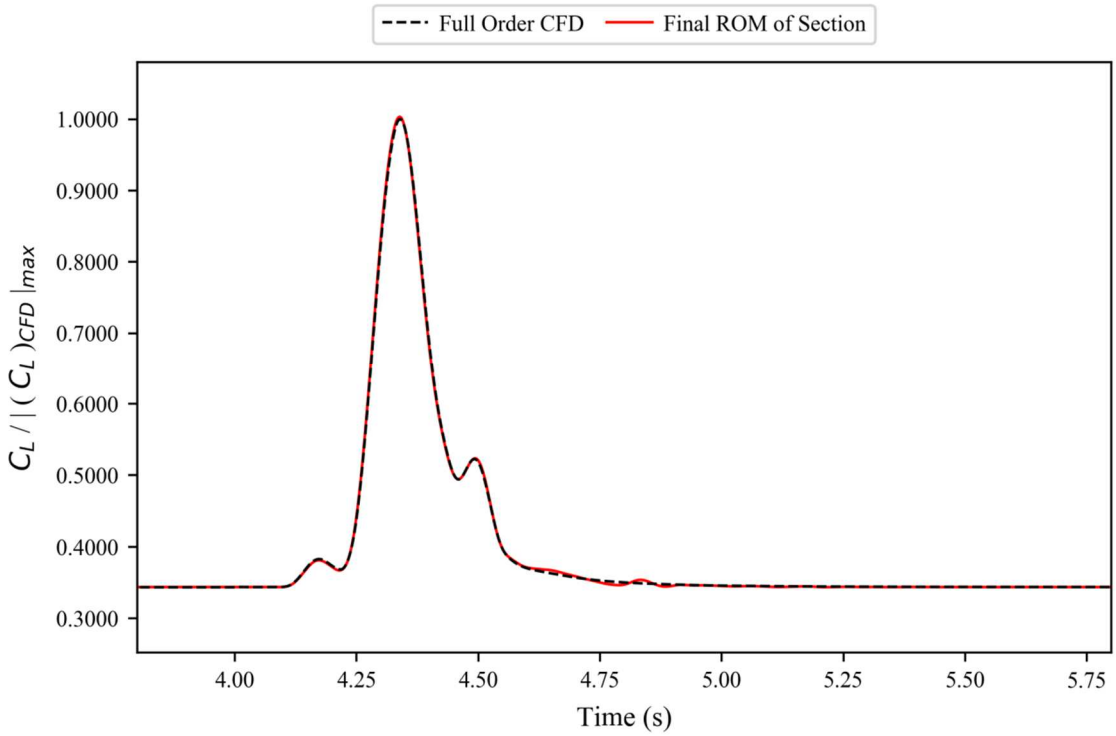

Figure 5.51. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 1, gust case 1.

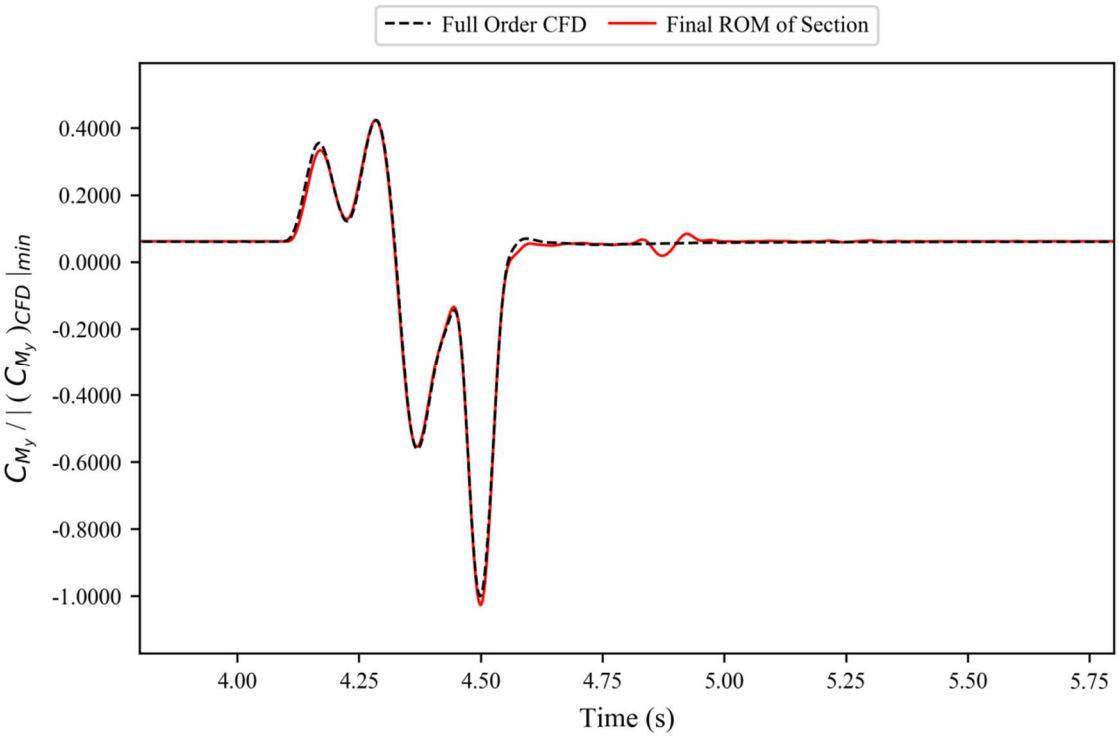

Figure 5.52. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 1, gust case 1.

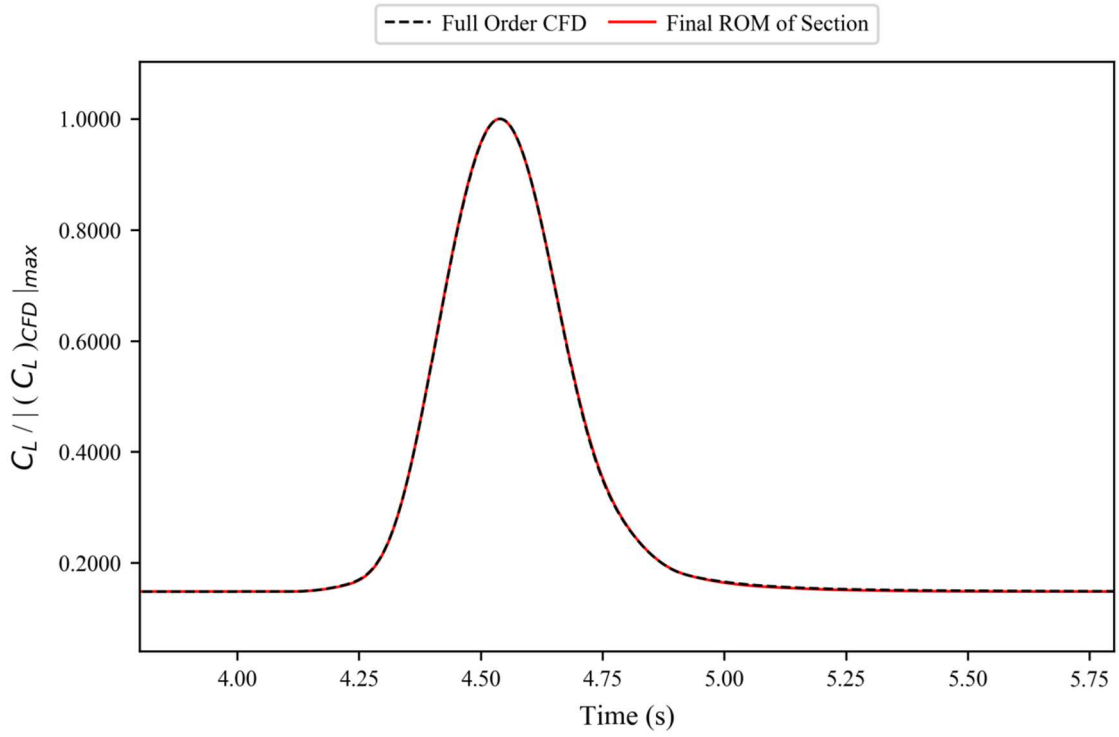

Figure 5.53. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 1, gust case 2.

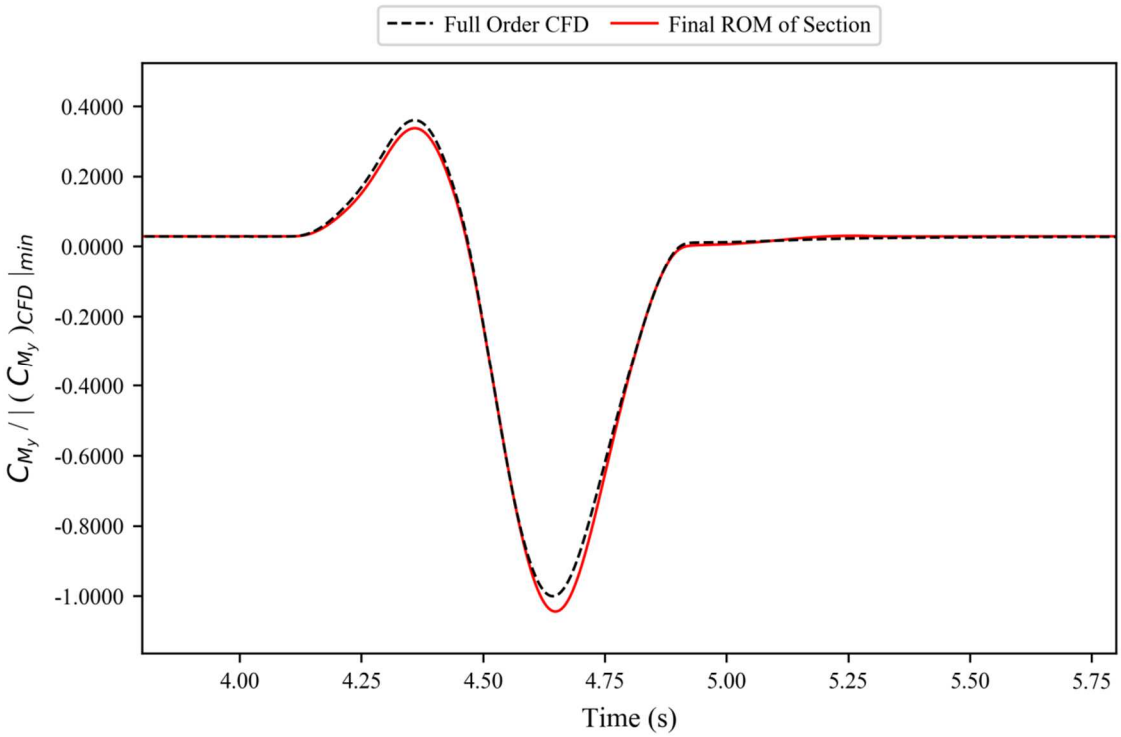

Figure 5.54. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 1, gust case 2.

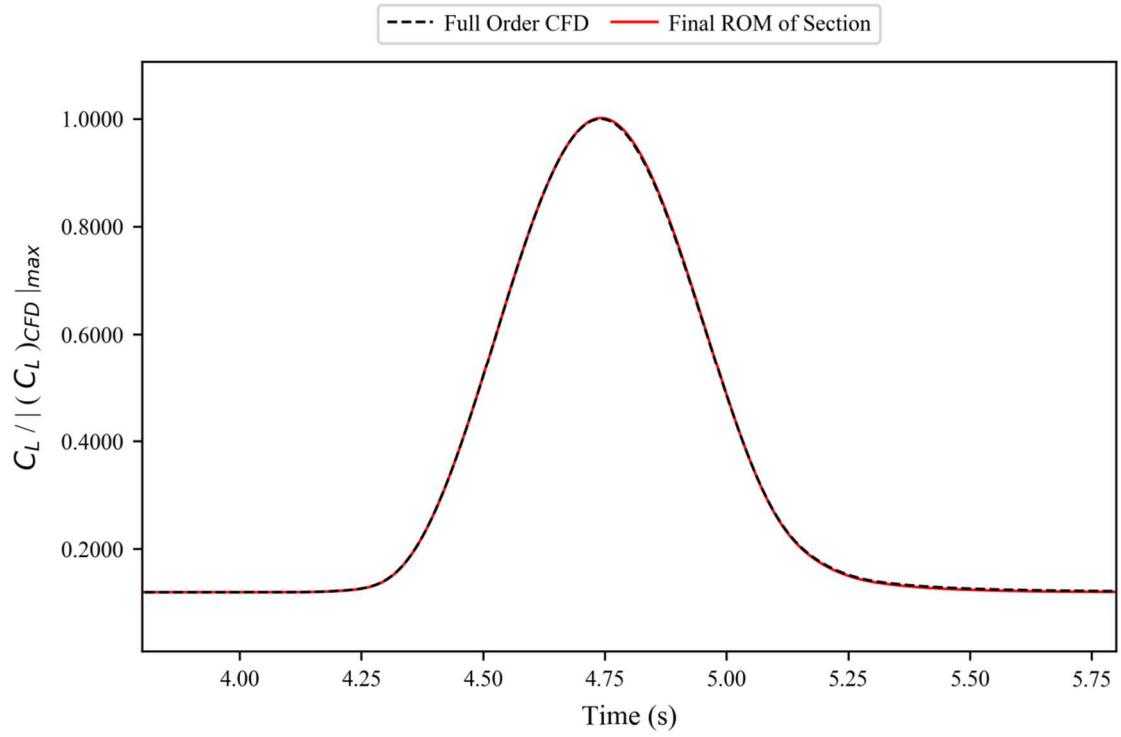

Figure 5.55. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 1, gust case 3.

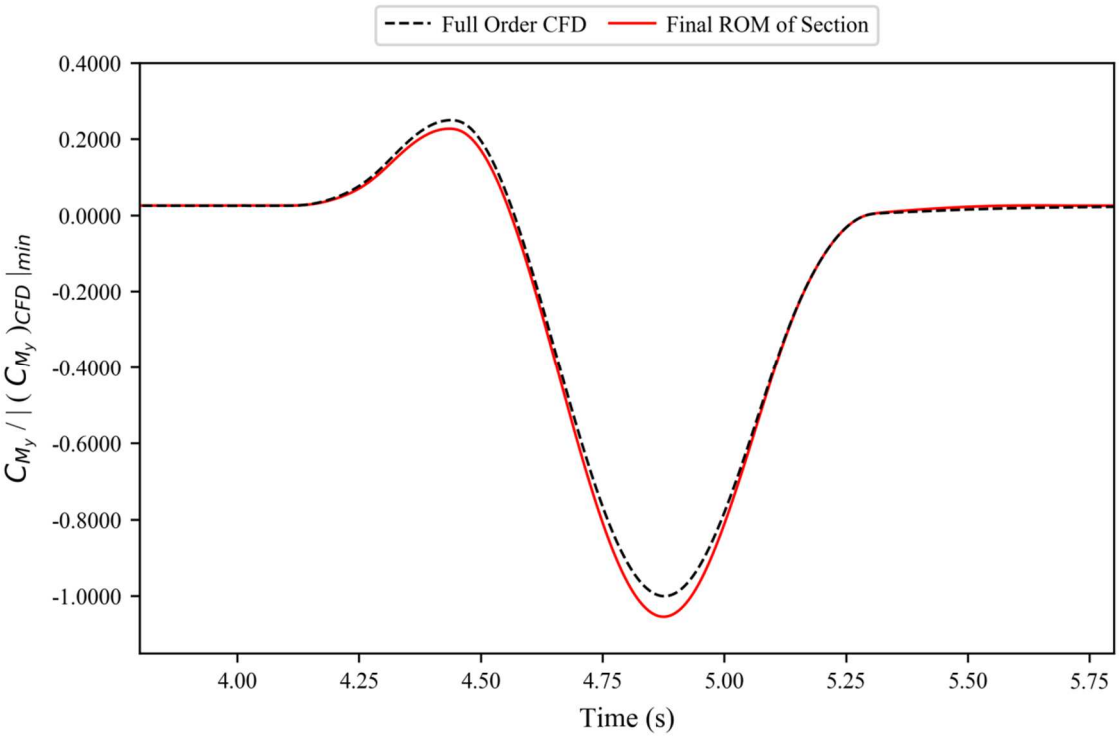

Figure 5.56. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 1, gust case 3.

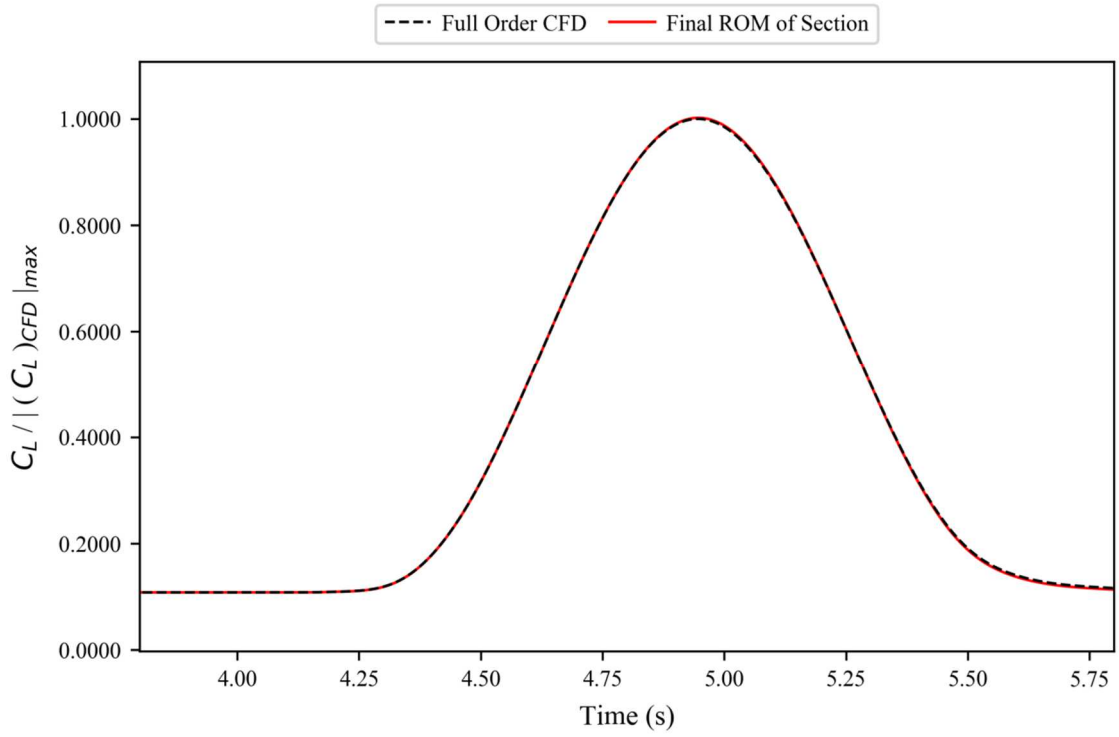

Figure 5.57. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 1, gust case 4.

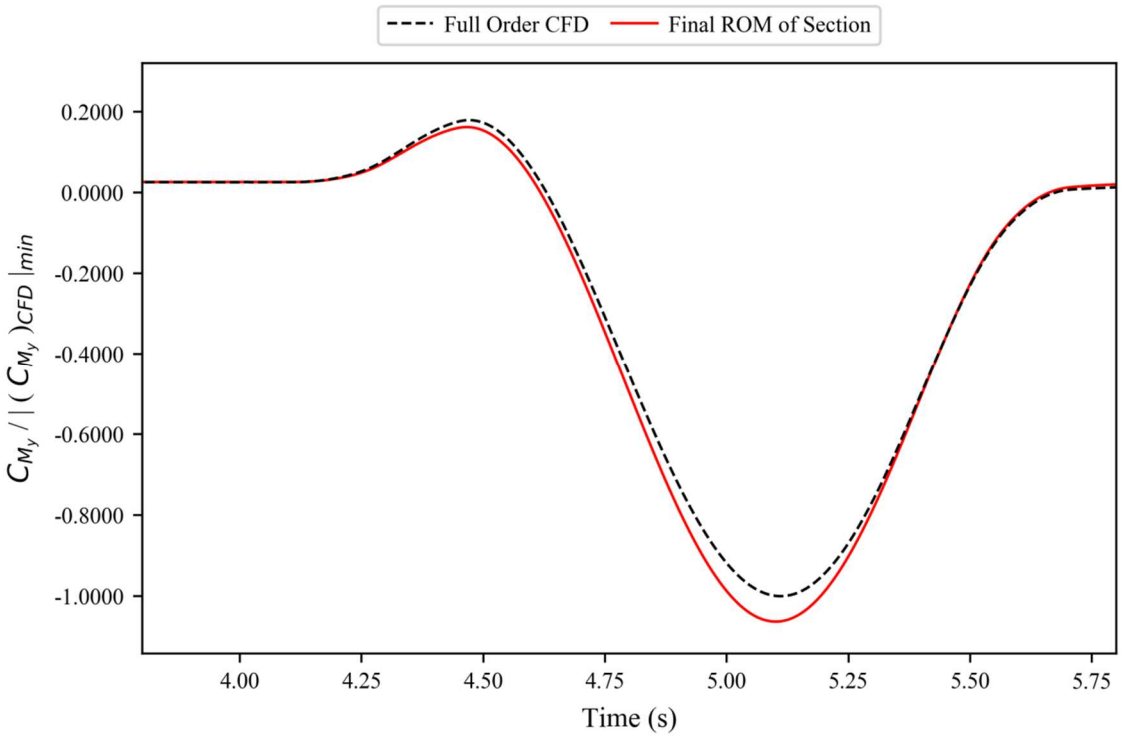

Figure 5.58. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 1, gust case 4.

5.4.2. Flight Point 3

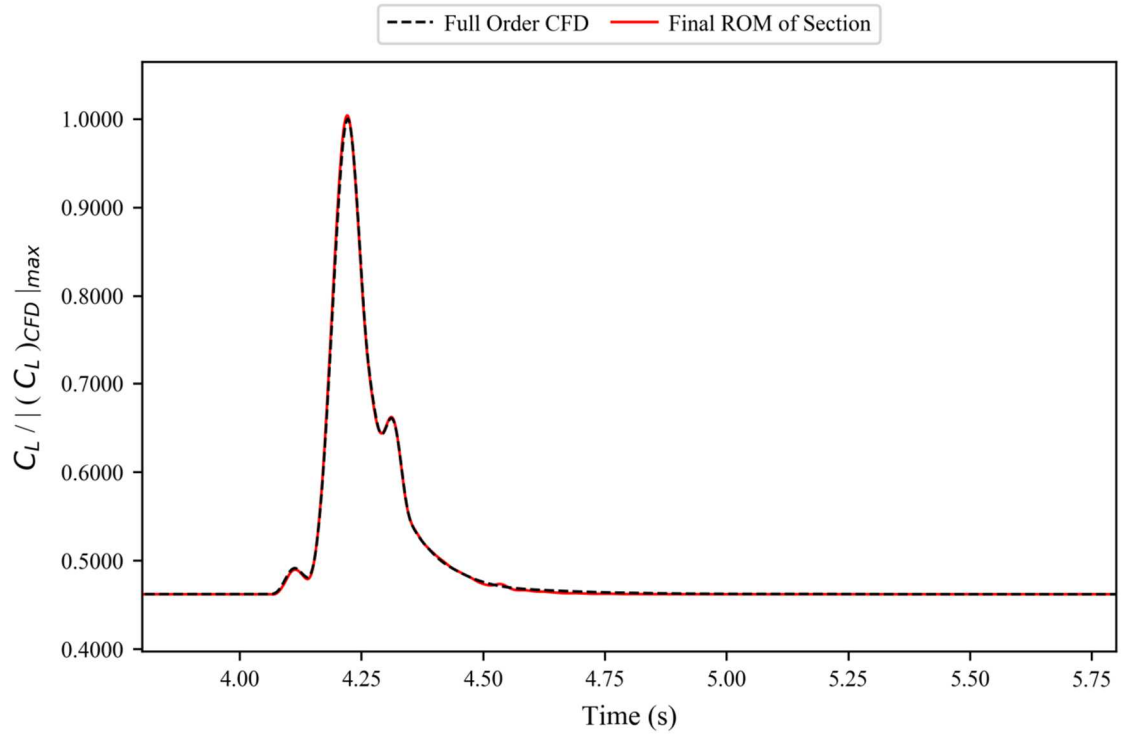

Figure 5.59. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 3, gust case 1.

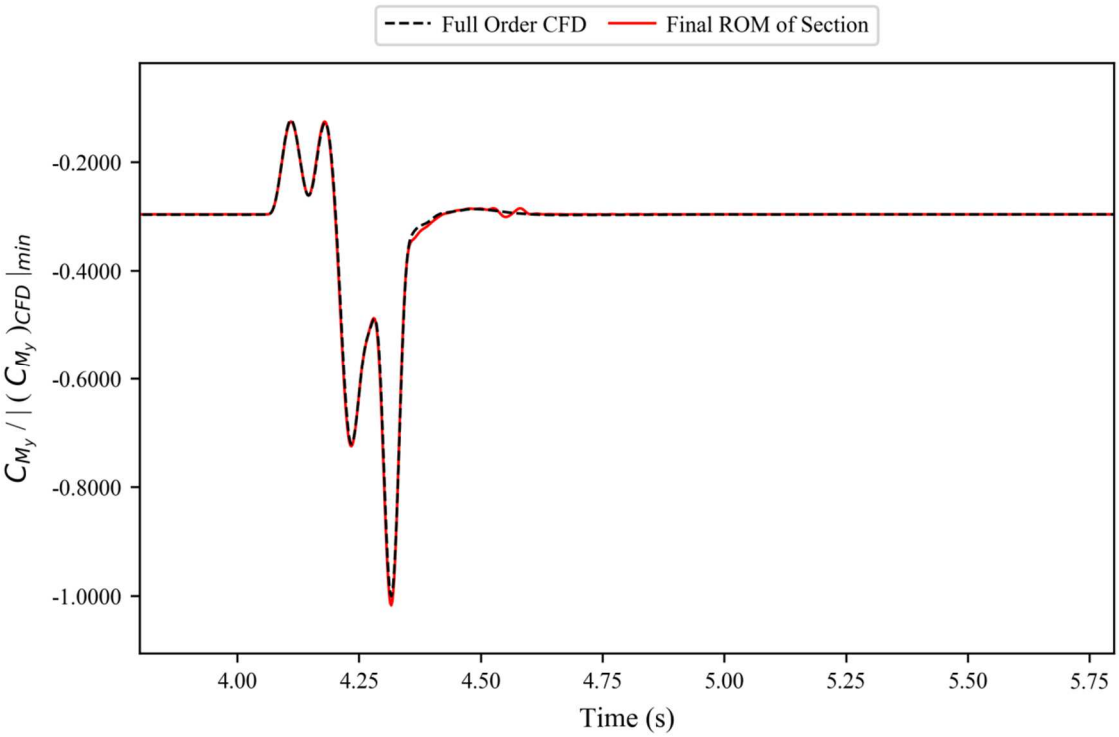

Figure 5.60. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 3, gust case 1.

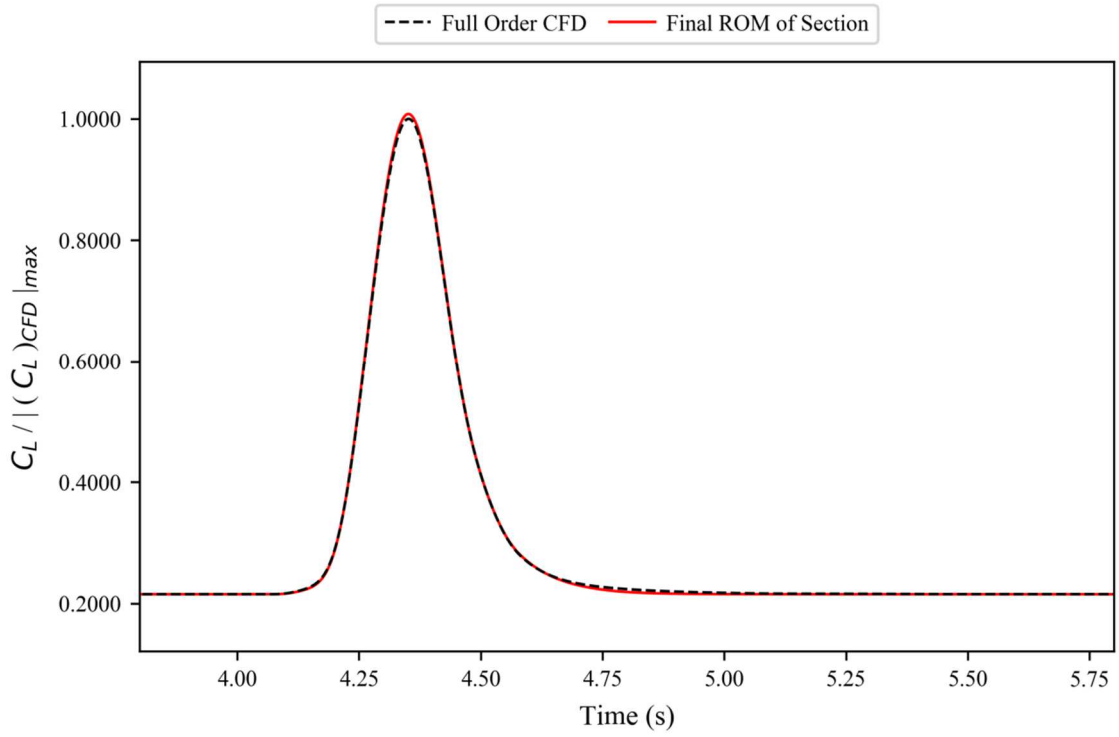

Figure 5.61. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 3, gust case 2.

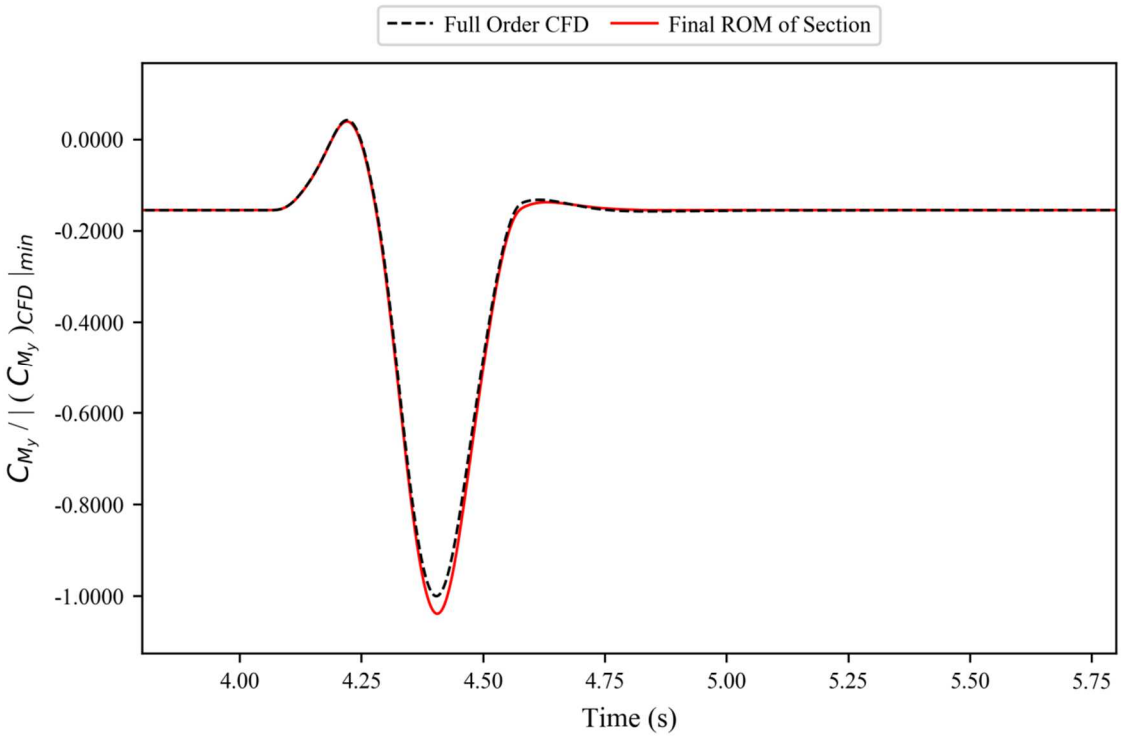

Figure 5.62. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 3, gust case 2.

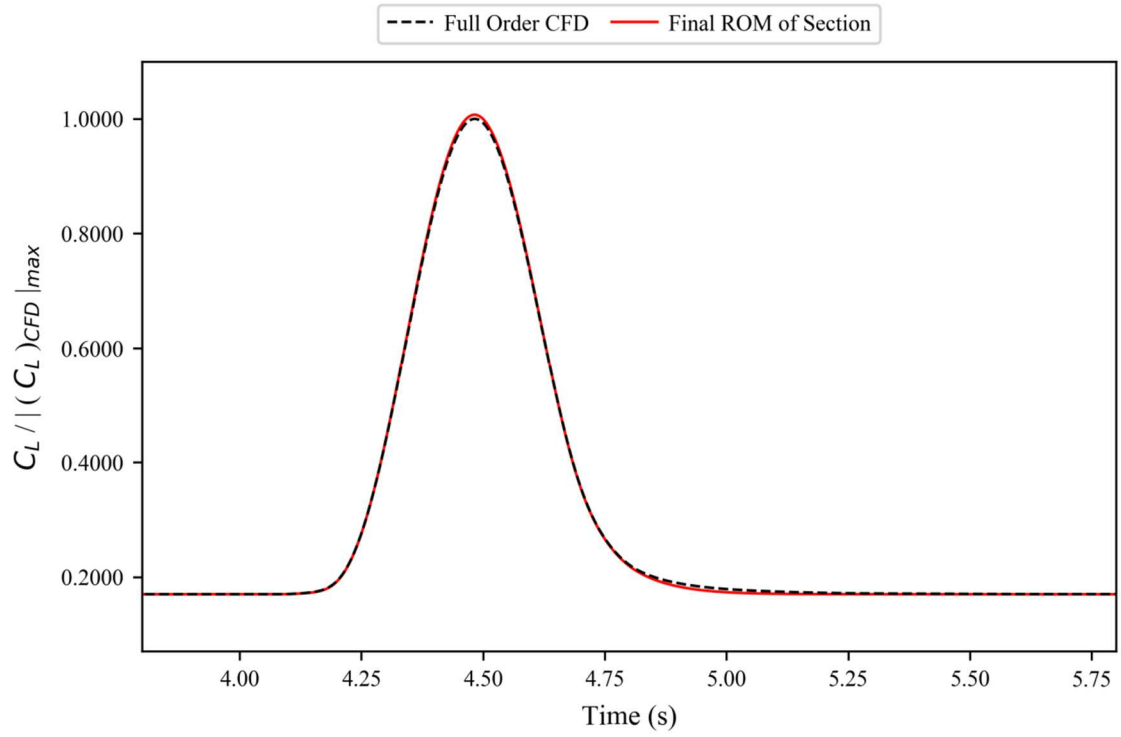

Figure 5.63. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 3, gust case 3.

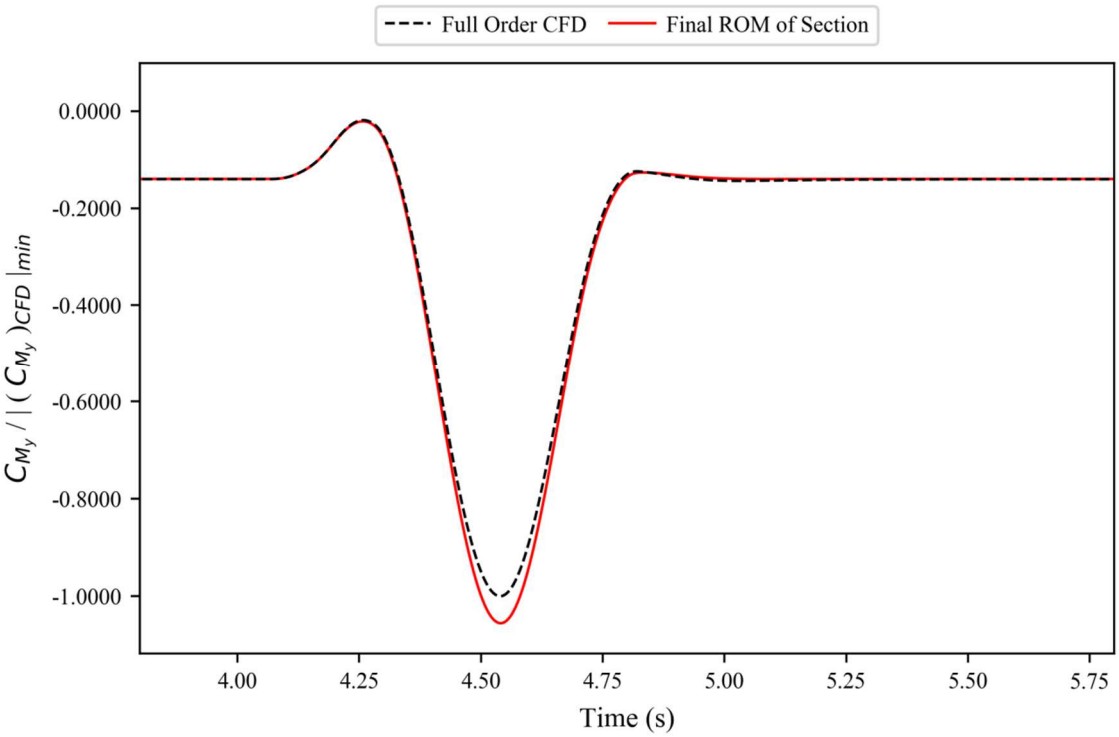

Figure 5.64. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 3, gust case 3.

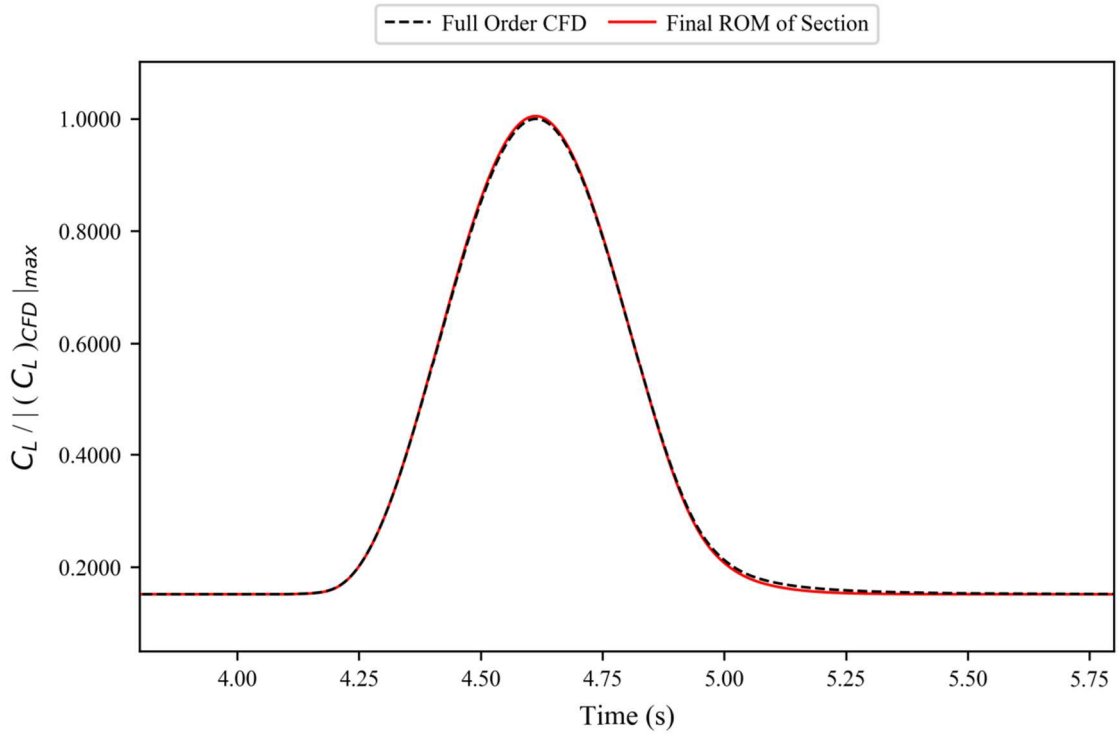

Figure 5.65. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 3, gust case 4.

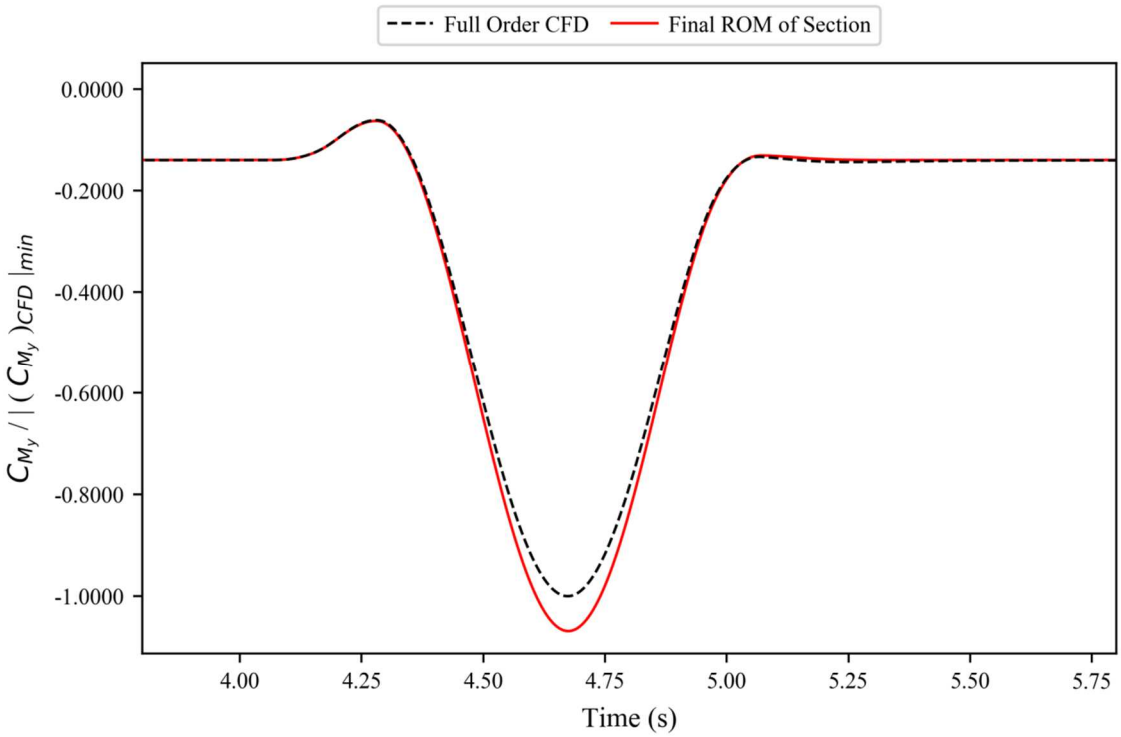

Figure 5.66. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 3, gust case 4.

5.4.3. Flight Point 4

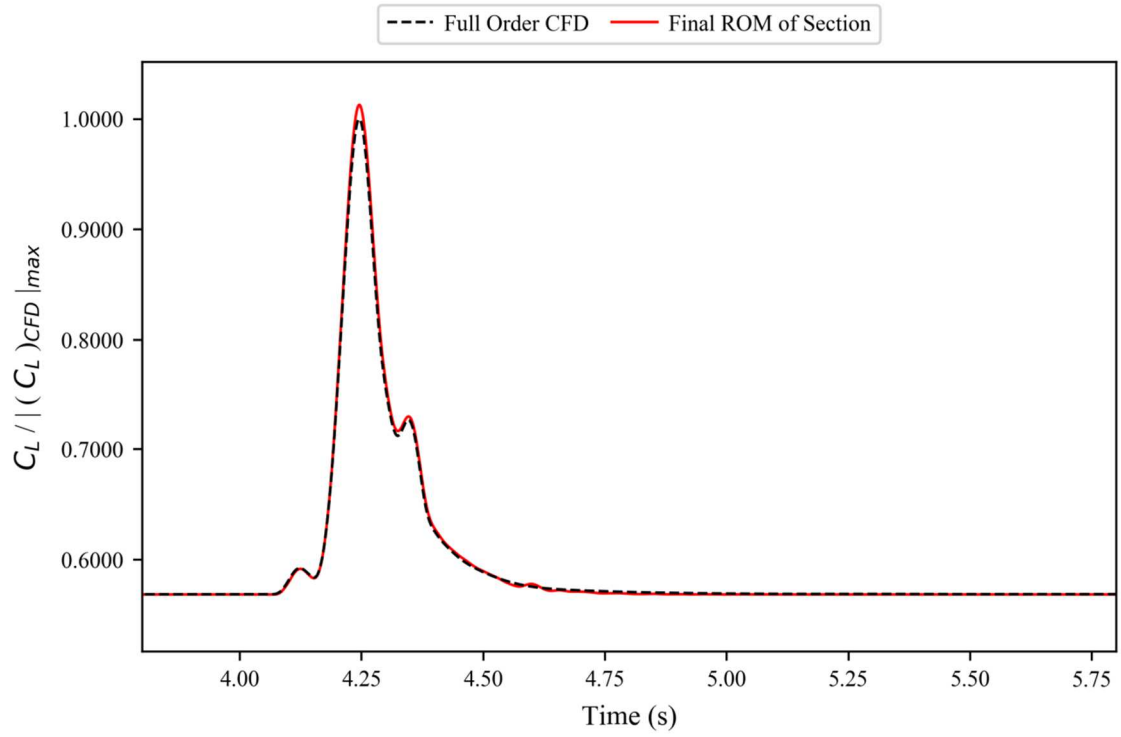

Figure 5.67. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 4, gust case 1.

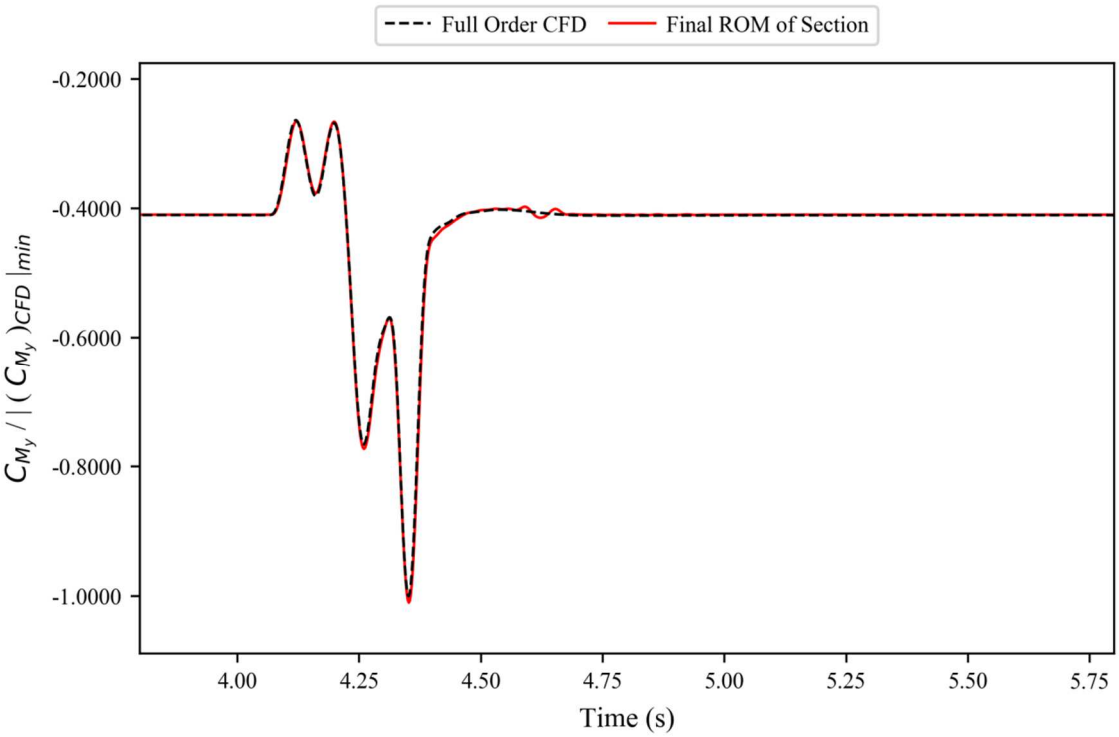

Figure 5.68. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 4, gust case 1.

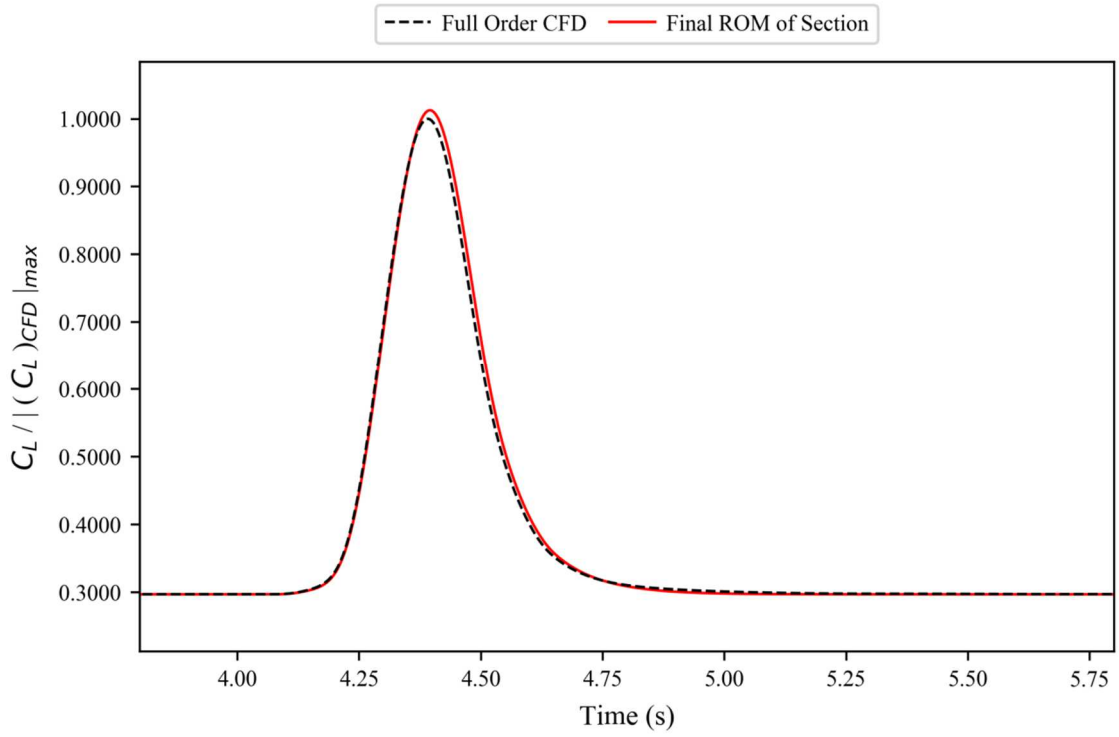

Figure 5.69. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 4, gust case 2.

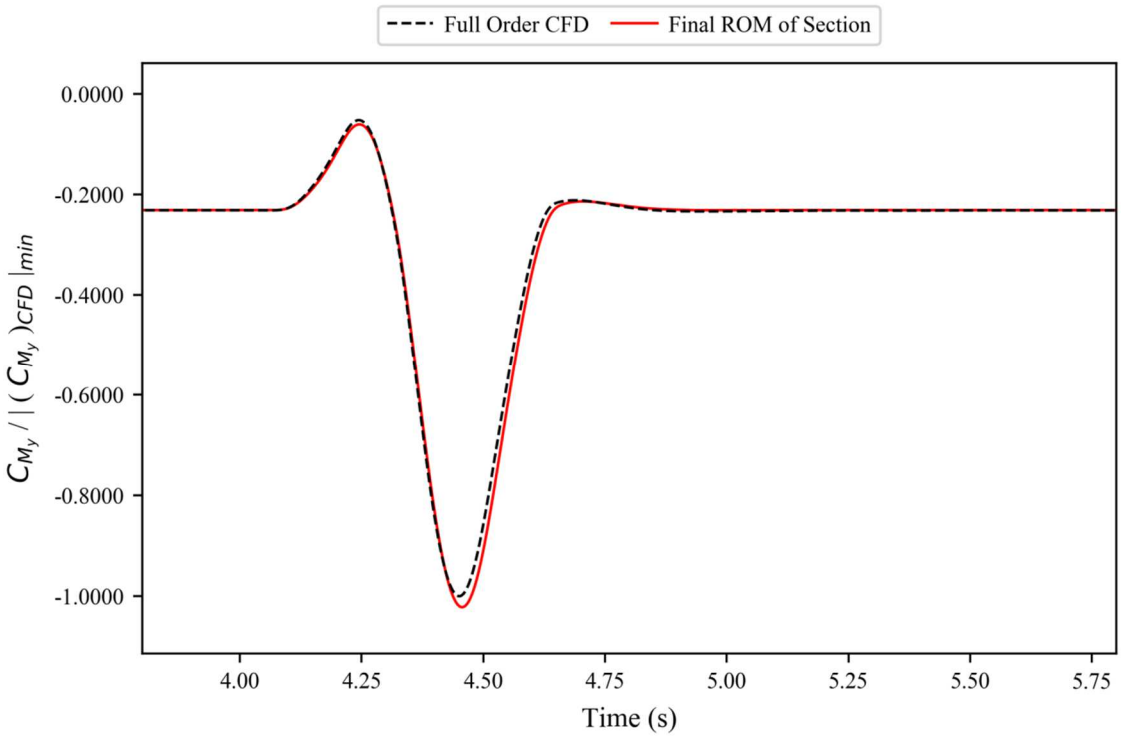

Figure 5.70. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 4, gust case 2.

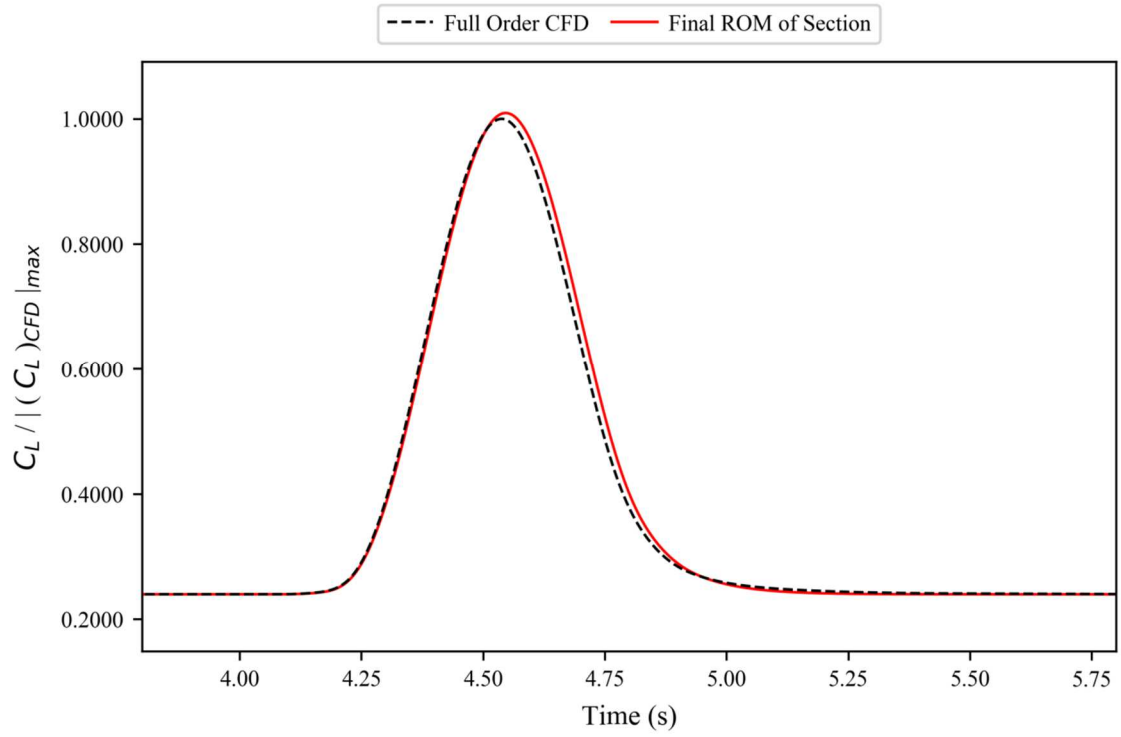

Figure 5.71. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 4, gust case 3.

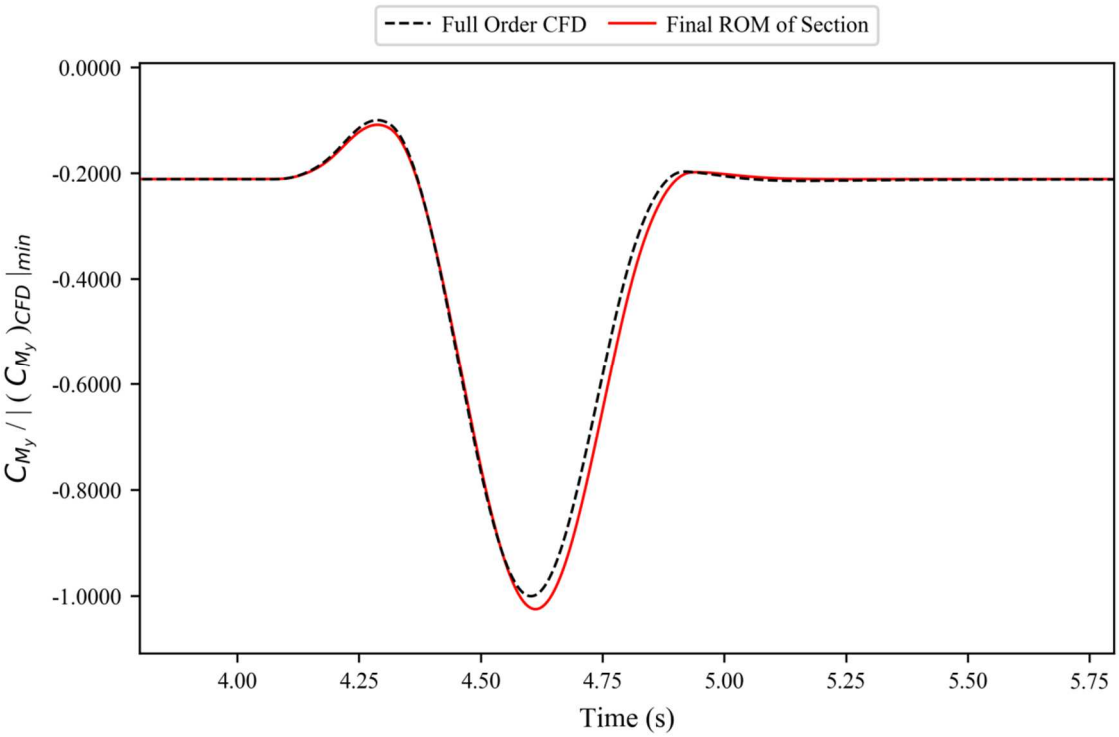

Figure 5.72. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 4, gust case 3.

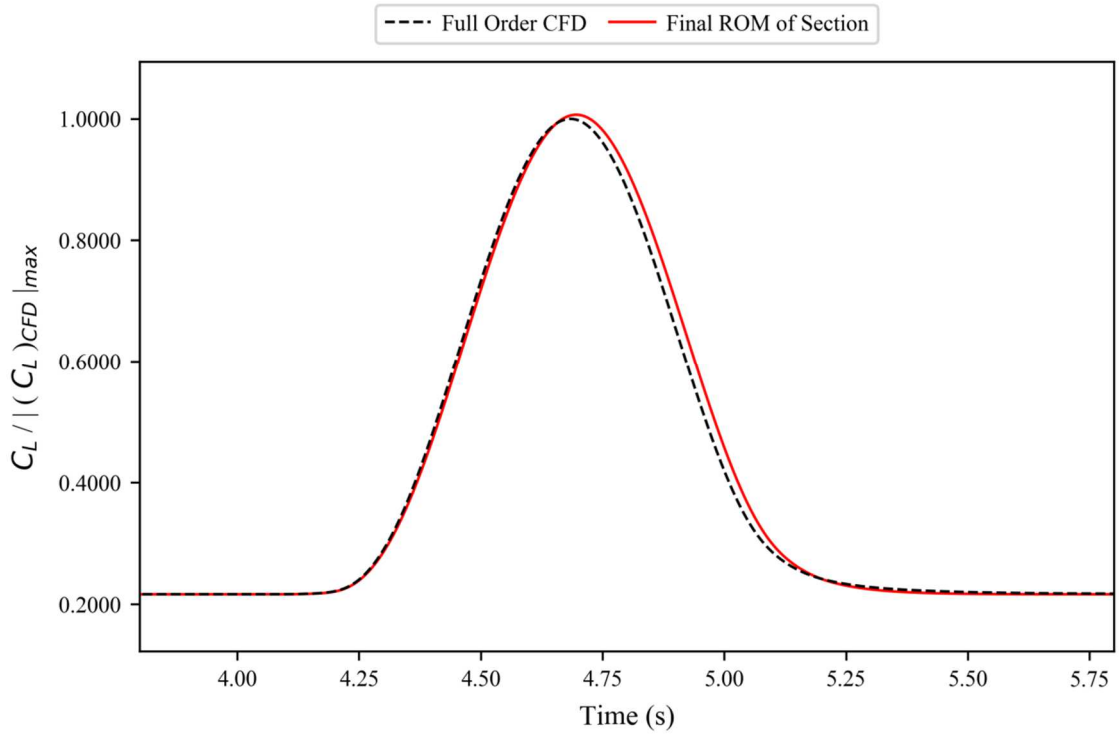

Figure 5.73. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 4, gust case 4.

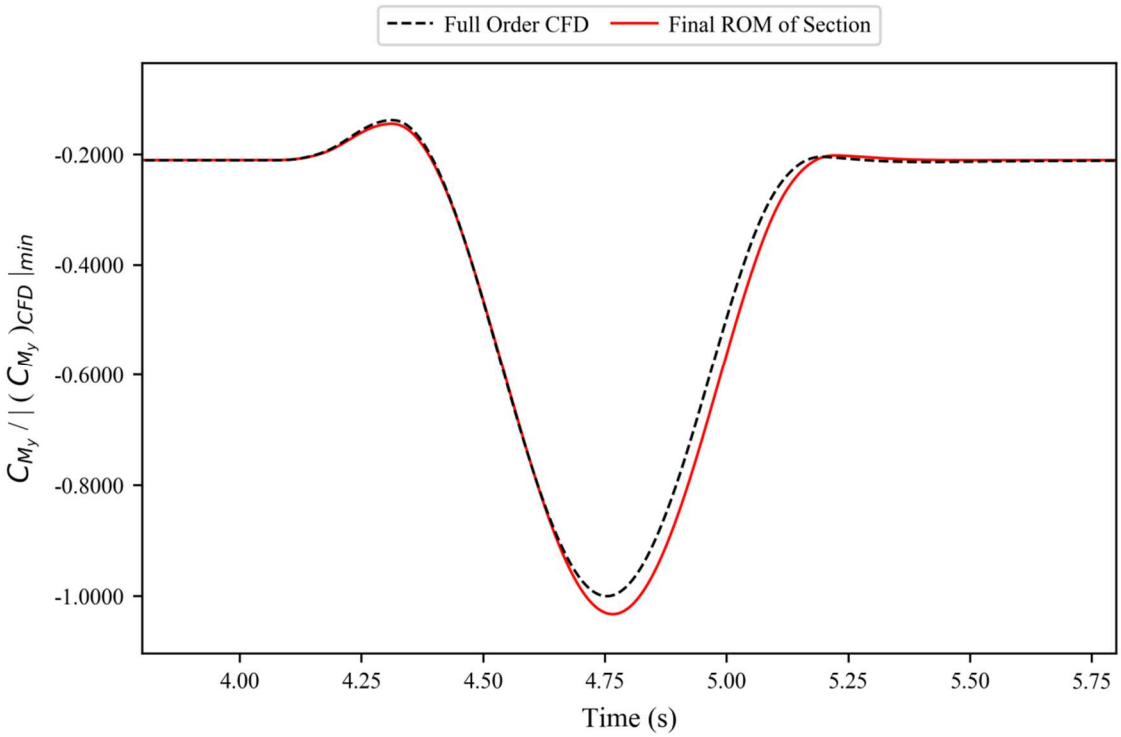

Figure 5.74. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 4, gust case 4.

5.4.4. Flight Point 5

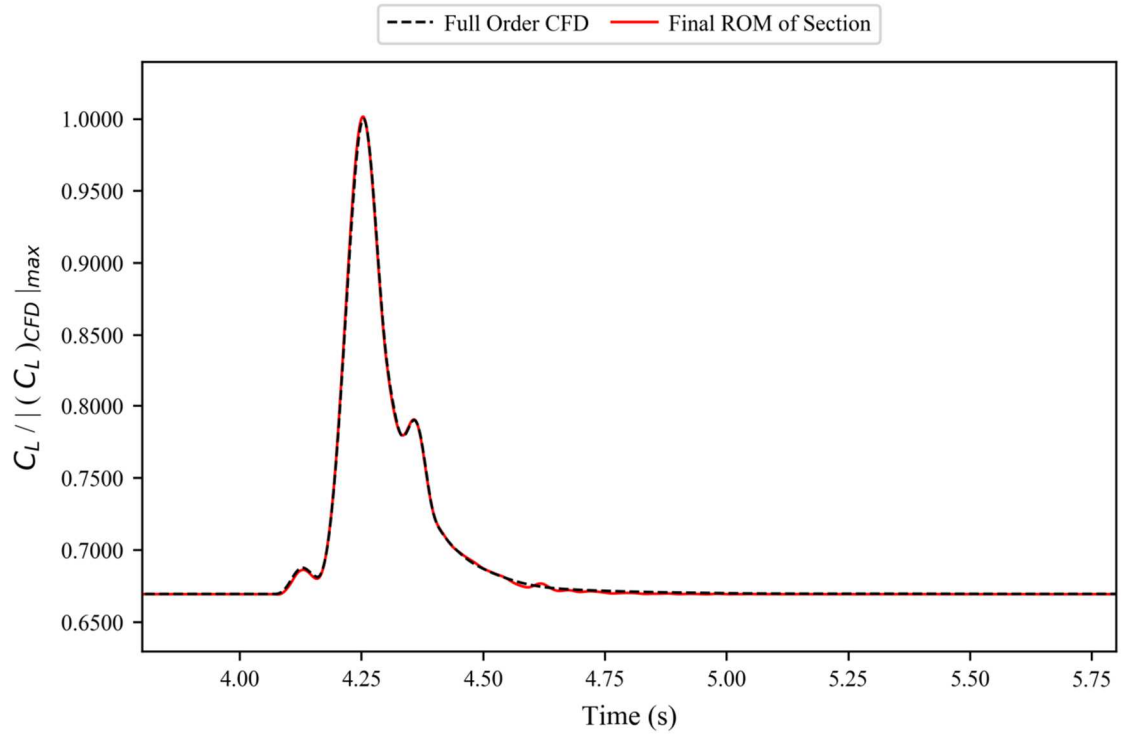

Figure 5.75. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 5, gust case 1.

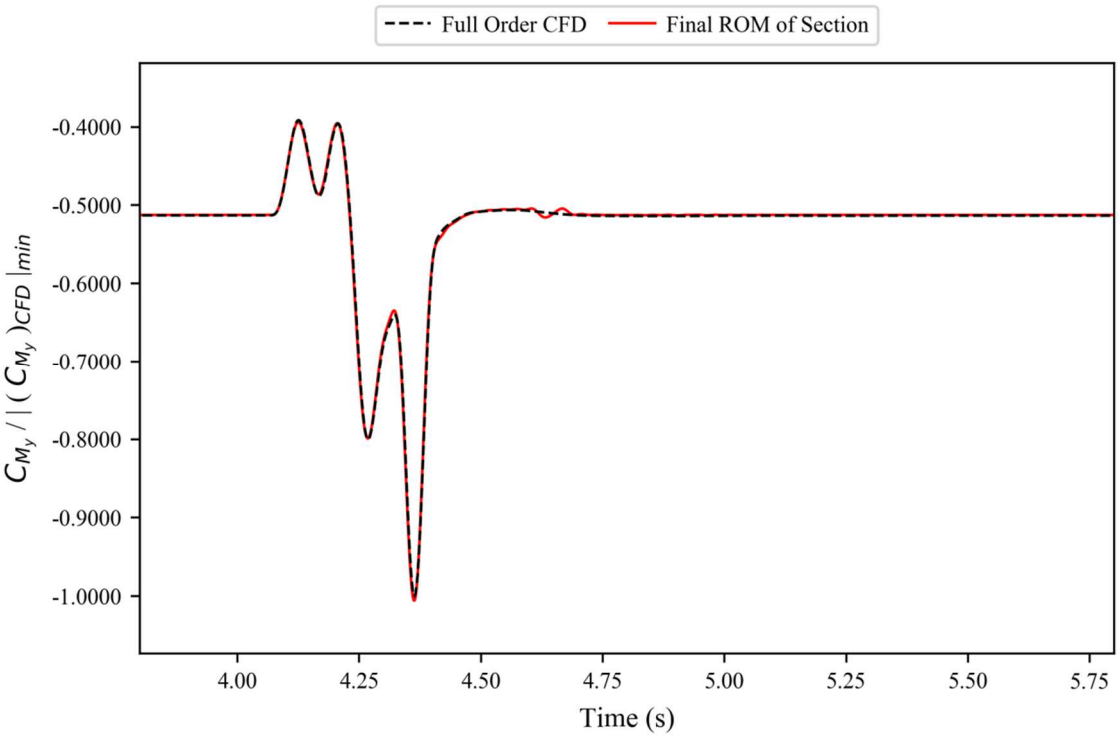

Figure 5.76. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 5, gust case 1.

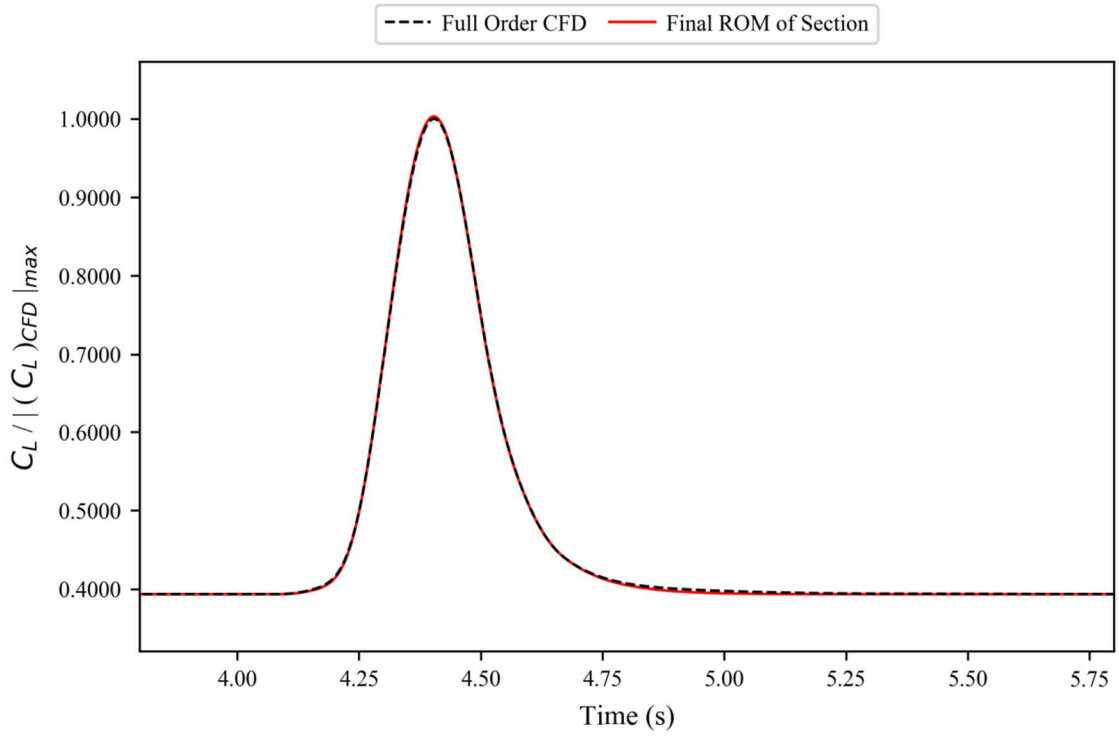

Figure 5.77. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 5, gust case 2.

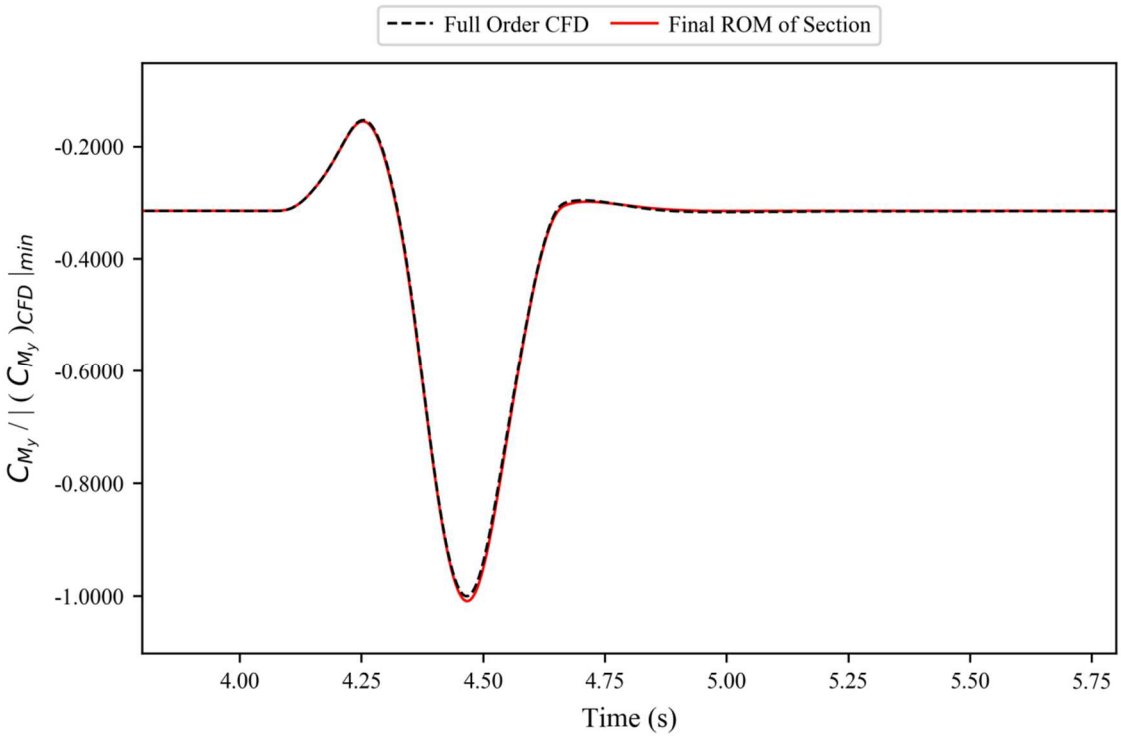

Figure 5.78. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 5, gust case 2.

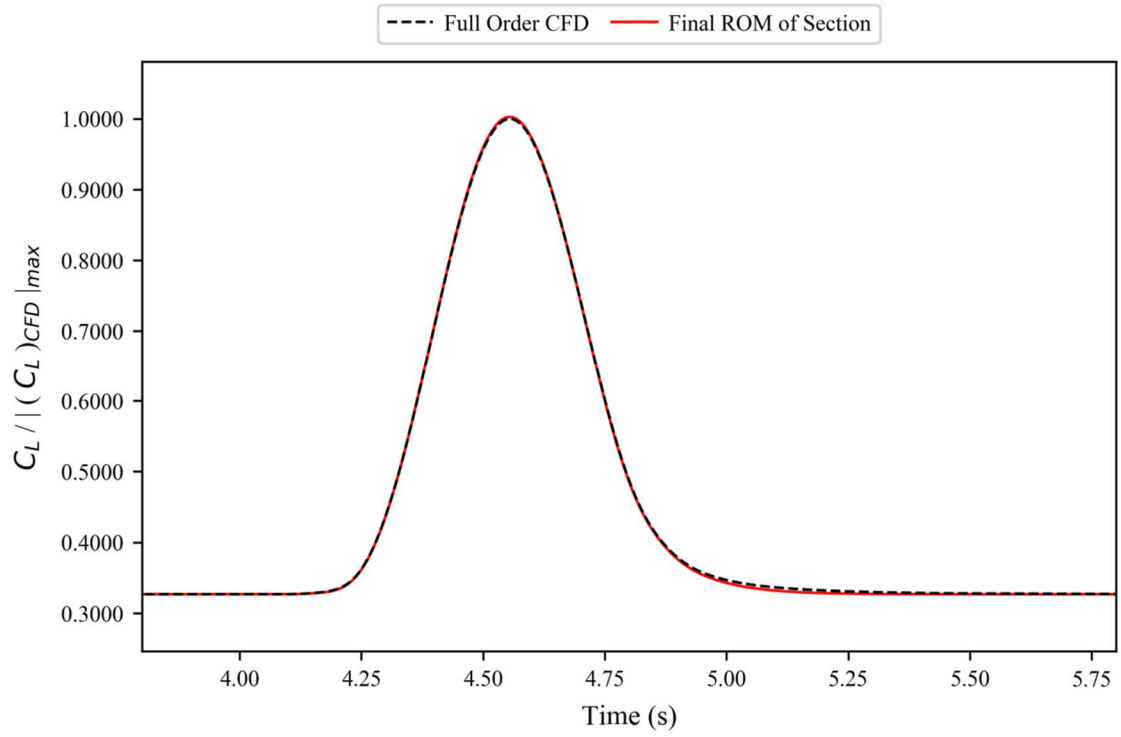

Figure 5.79. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 5, gust case 3.

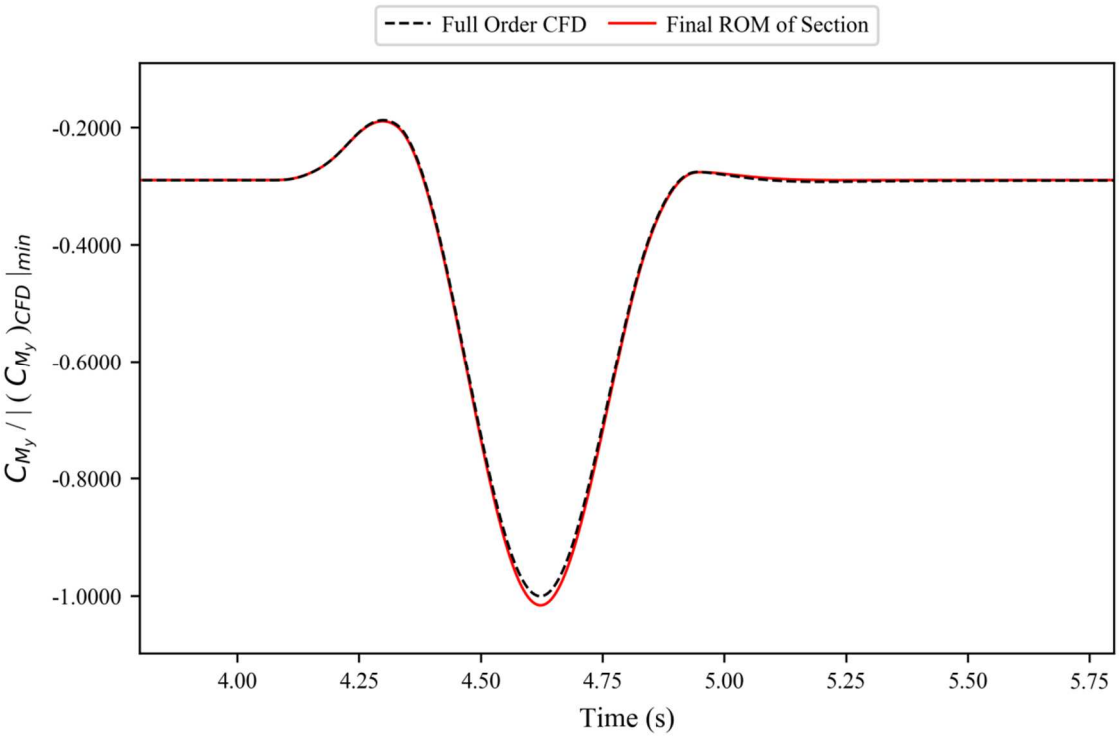

Figure 5.80. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 5, gust case 3.

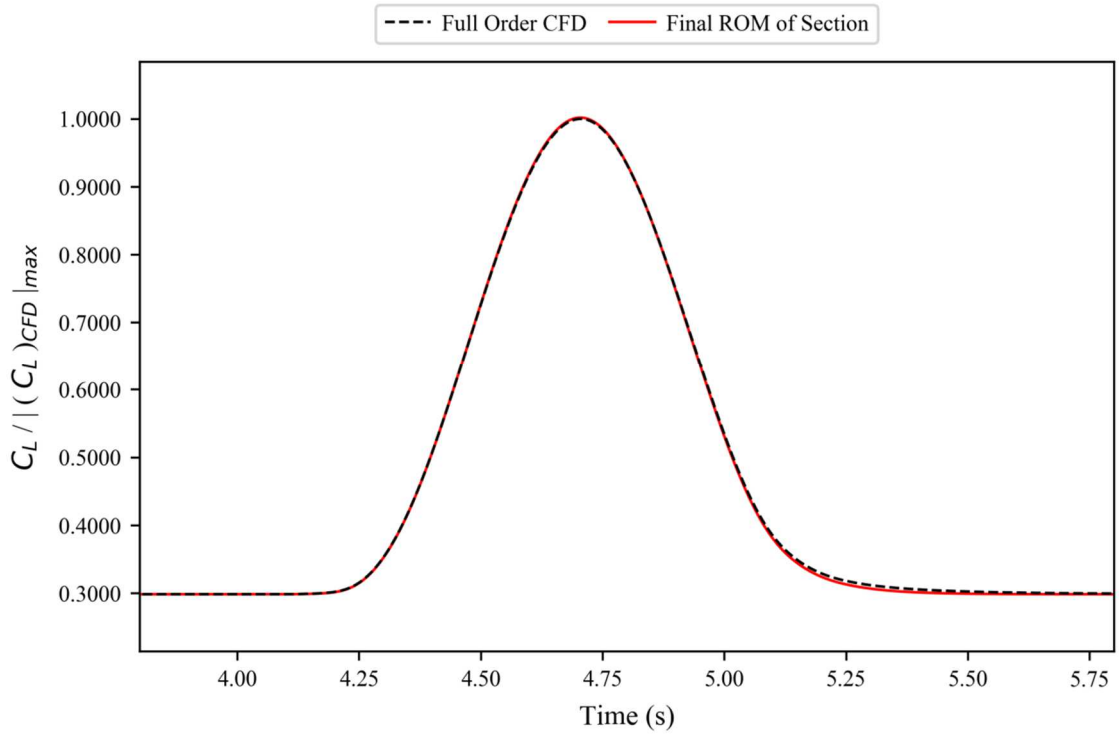

Figure 5.81. ROM results against full order CFD simulation for the, normalised, coefficient of lift for a generic wide bodied aircraft at flight point 5, gust case 4.

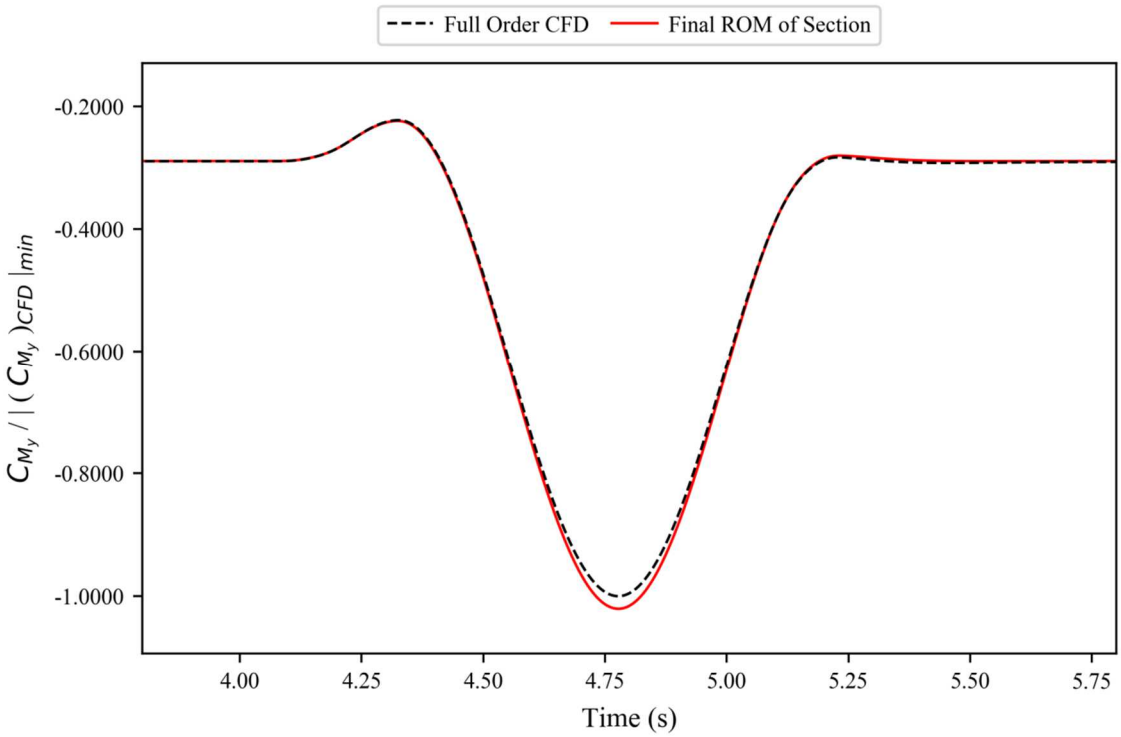

Figure 5.82. ROM results against full order CFD simulation for the, normalised, coefficient of pitching moment for a generic wide bodied aircraft at flight point 5, gust case 4.

5.4.5. Conclusions

All the results (Figures 5.18-5.23 and 5.51-5.82) generally show a very high level of accuracy, with the only noteworthy discrepancy being a small inaccuracy at the peaks of some of the gusts; in particular, the longer gusts and particularly flight point 3. This slight degradation in accuracy can be explained by looking at the coefficients of pressures on the top surfaces of the aircraft for flight point 3, both in steady conditions (Figure 5.83) and at the peak of gust case 4 (Figure 5.84); from which it can be seen that the gust appears to cause a leading edge separation to occur near the wing root. As a result, the response is non-linear, which explains the slight degradation in ROM accuracy.

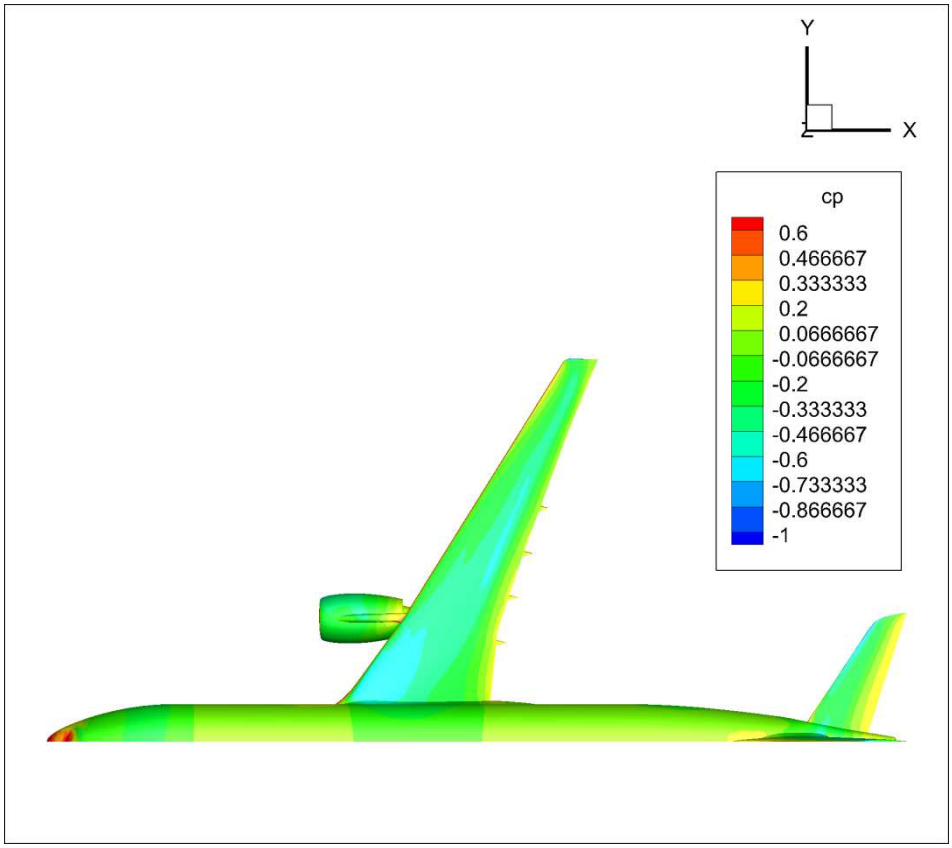

Figure 5.83. Coefficients of pressure on the top of the whole aircraft model from a steady simulation at flight point 3.

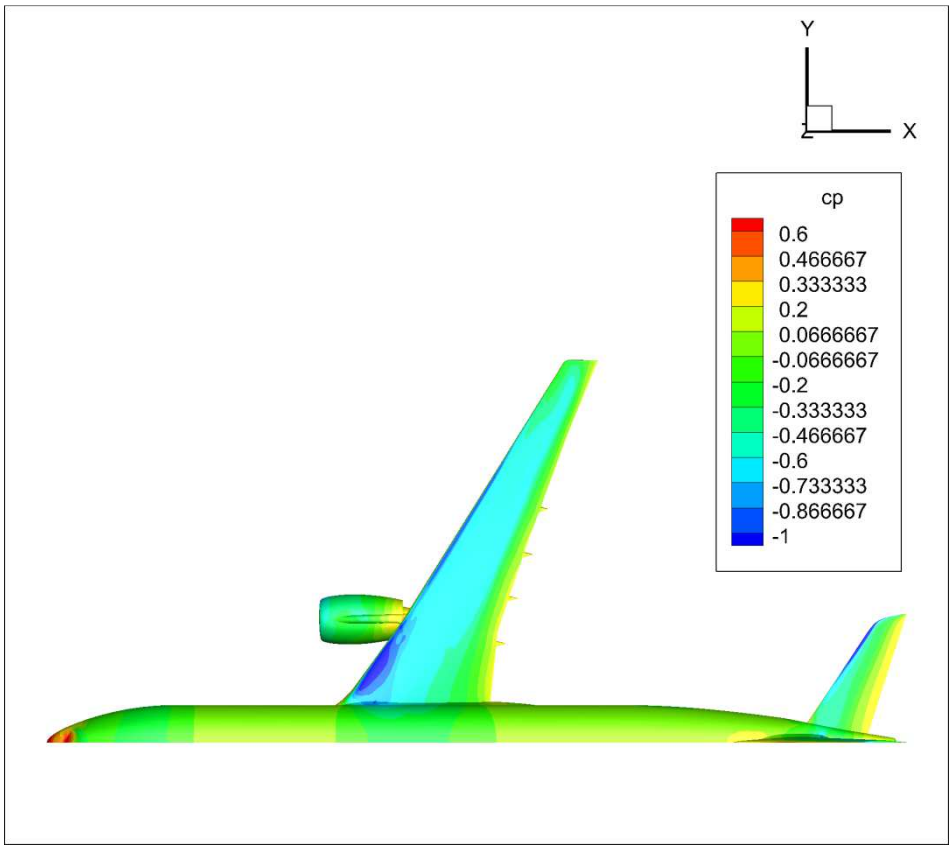

Figure 5.84. Coefficients of pressure on the top of the whole aircraft model at the peak of gust case 4 for flight point 3.

However, despite the slight degradation in the accuracy of the ROM in the non-linear cases, the overall high level of accuracy highlights that the final ROM methodology put forward at the end of Section 4.6 was also suitable for this viscous, more complex geometry model with minimal modification; the only change being the alternative stabilisation method. A ROM has been produced that balances robustness, accuracy and computational cost, as tested on a more industrial representative model.

6. ROM Applied to Simplified Aircraft Model Including Flight Mechanics

All test cases considered so far have been rigid, and so to further test the capabilities of the ROM method, an aircraft model that includes aeroelastic effects and which has six rigid-body degrees of freedom is considered. A brief overview is provided as the author was not involved in the development of the elastic and flight mechanics model.

The aircraft model is simplified to model only the wing and empennage. Unlike the previous models, this one is full width, and so there is no symmetry plane artificially added; as shown in Figure 6.1.

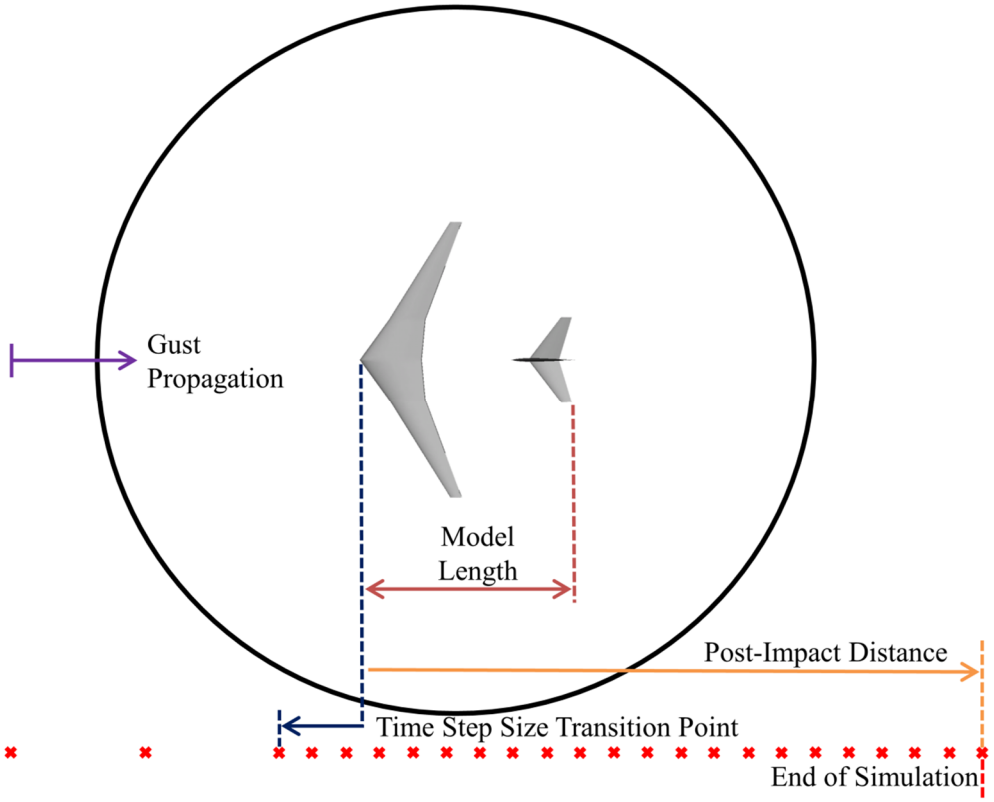

Figure 6.1. Representation (not to scale) of the top down view of the domain for the simplified aircraft geometry.

The aerodynamic model consisted of an unstructured mesh throughout a hemisphere domain (see Figures 6.2-6.4). The mesh was made up of 796,806 tetrahedron cells and had 52,276 triangular surface elements. As these were inviscid simulations, the Euler equations acted as the governing equations being solved for (see Section 2.3).

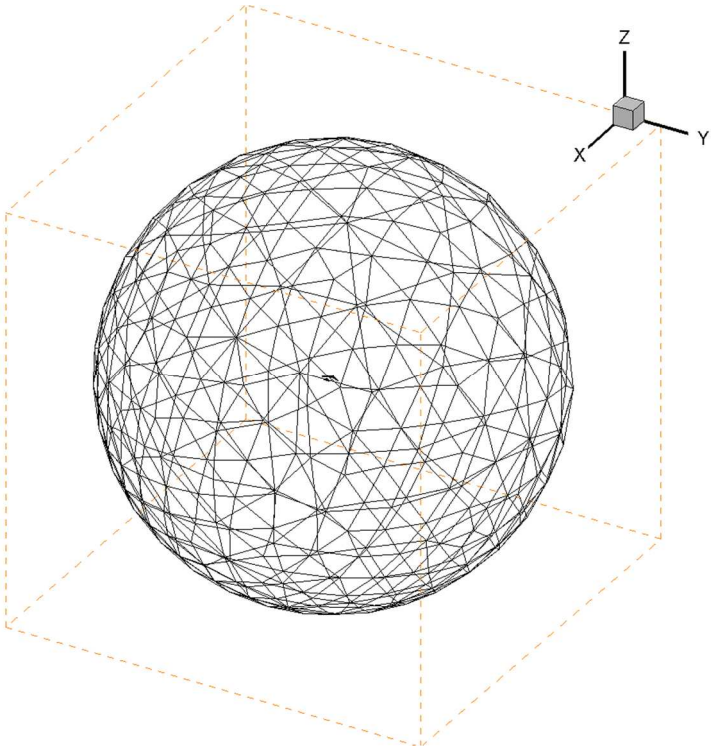

Figure 6.2. Domain mesh for the simplified aircraft model.

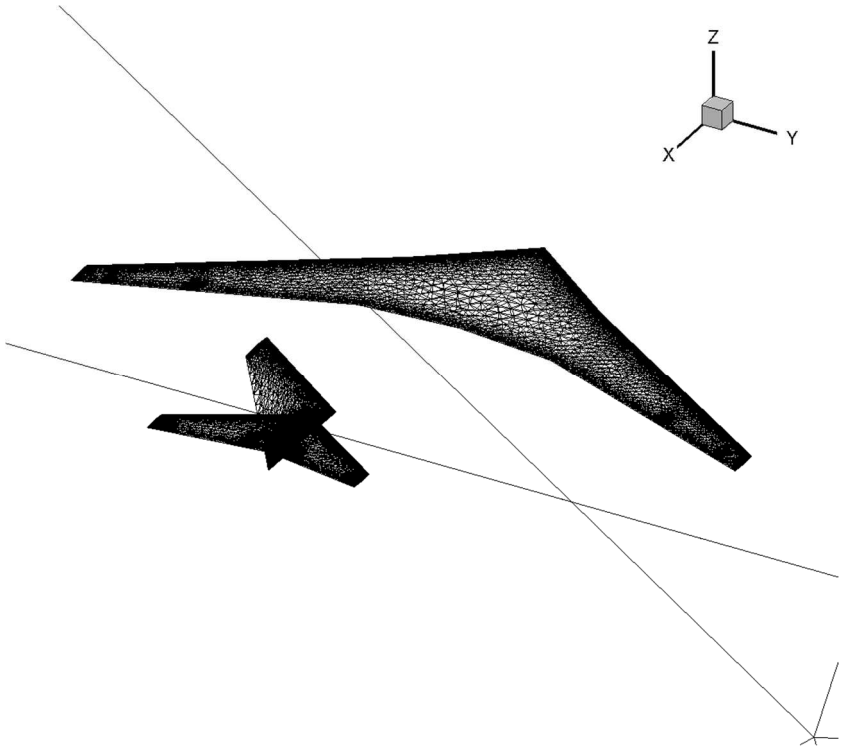

Figure 6.3. Surface mesh for the simplified aircraft model.

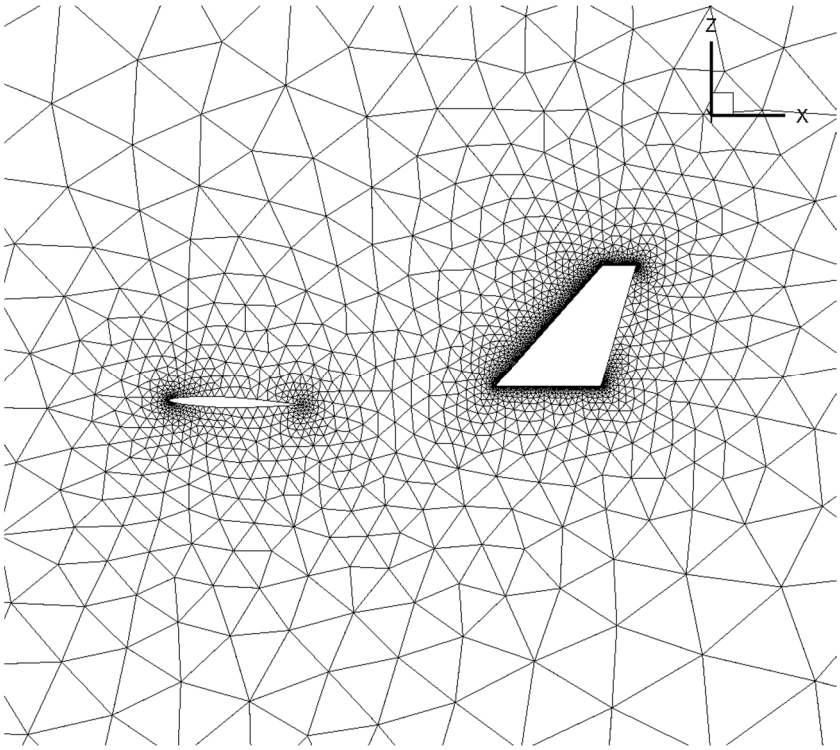

Figure 6.4. Mesh on slice through the centre of the simplified aircraft model.

The aeroelastic model consisted of a simple, beam stick model; implemented within the Airbus/DLR developed program Flow Simulator; which for Flight Point 3 (see Table 6) resulted in a tip deflection of approximately 0.4m during steady simulations, which rose to approximately 0.55m at the peak of the 18m gust. The flight mechanics was a 6 degrees of freedom model, with masses and inertias taken from a modified A350 model. The six rigid-body degrees of freedom are movement in any of the 3 primary axes, as well as pitch, yaw and roll. The horizontal stabiliser has moveable elevators, and as such the aircraft can be trimmed for straight and level flight; however, once trimmed the elevators were kept fixed during the gust encounter.

For this model, no gust alleviation factor was used and as in Chapters 0 and 0, five flight points were selected, each with four gust cases; as seen in Table 6.

Table 6. Details of gusts explored for the flight mechanics model.

Flight Point	Mach Number	Altitude (ft)	Altitude (m)	Gust 1 18m Velocity (ms^{-1})	Gust 2 80m Velocity (ms^{-1})	Gust 3 146m Velocity (ms^{-1})	Gust 4 214m Velocity (ms^{-1})
1	0.500	0	0	11.299	14.488	16.016	17.070
2	0.735	0	0	11.299	14.488	16.016	17.070
3	0.800	0	0	11.299	14.488	16.016	17.070
4	0.800	29,9995	9,142.476	9.684	12.417	13.727	14.630
5	0.800	43,000	13,106.400	7.321	9.388	10.378	11.061

6.1. Length of Simulations

In Sections 4.4 and 5.2 it was shown that 2.33 model lengths (post-impact) balanced computational cost with accuracy. However, this was for two test cases with no rigid-body degrees of freedom. The introduction of these degrees of freedom are likely to have a notable impact as the settling of the system response after a step input excitation is likely to be far more dynamic. Therefore, it is necessary to consider again for how long (post-impact) the sharp-edged gust needs to be run.

As previously, the data from a single sharp-edged gust was artificially cut when the gust reached different locations downstream (measured in model lengths post-impact). These sharp-edged gusts of various lengths were then used to construct a ROM in the same manner as before. However, unlike the previous test cases, a ROM-size of 60 rather than 20 was used; this was because the system is expected to have a more complex response.

6.1.1. Results

The ROM method was tested on the gusts of flight point 2. Figures 6.5-6.12 show that the ROMs built with a 4.00 model length sharp-edged gusts produces poor results, with those built with 6.00 and 8.00 being the first that start to capture the system relatively well. However, it is not until 10.00 model lengths that the long term settling of the system is captured well; and therefore this is the logical choice to take forward.

It is worth noting that the results in general capture the system well, but are less accurate than those presented in Chapters 0 and 0. This is because the response is much less linear, and as the underlying method of constructing the ROM works on the concept of linearising the system response, as the non-linearities grow the accuracy is likely to suffer.

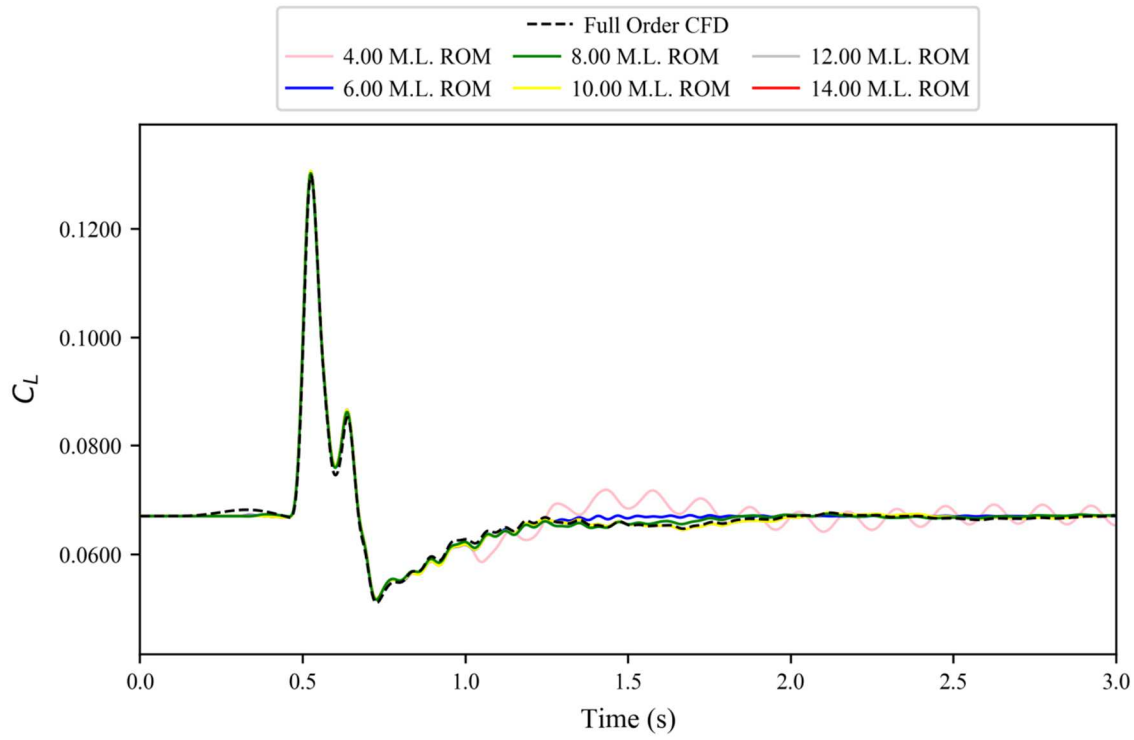

Figure 6.5. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of lift for a flight mechanics enabled model at flight point 2, gust case 1.

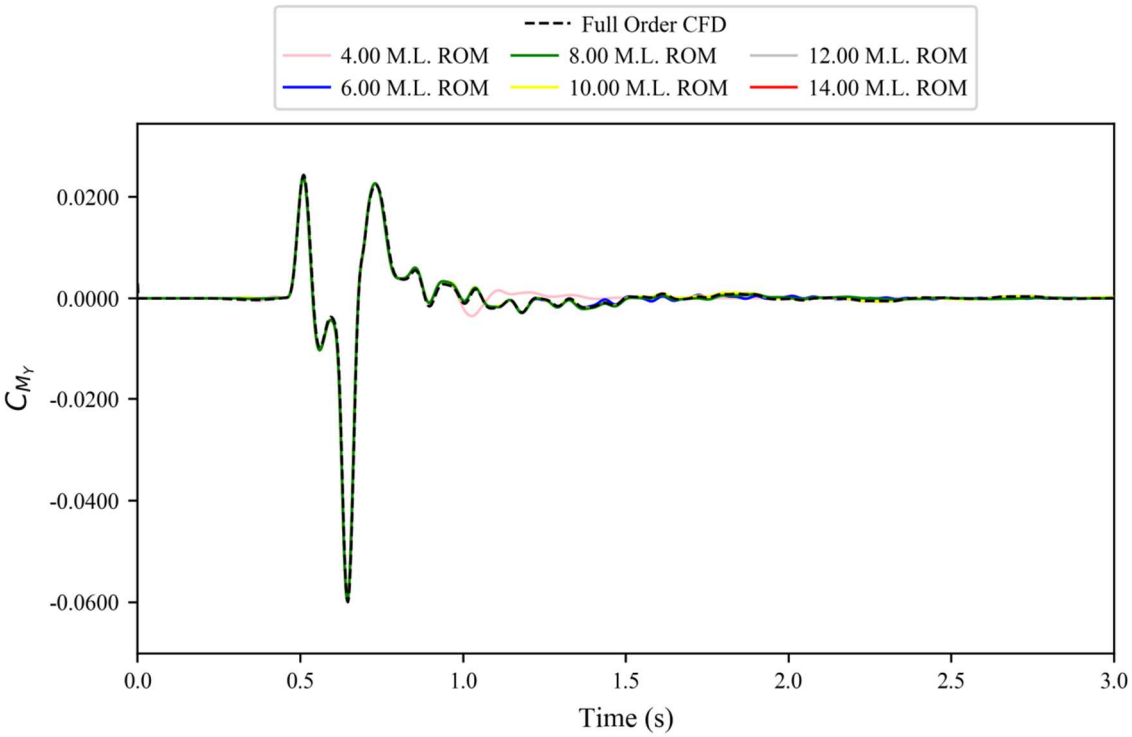

Figure 6.6. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 1.

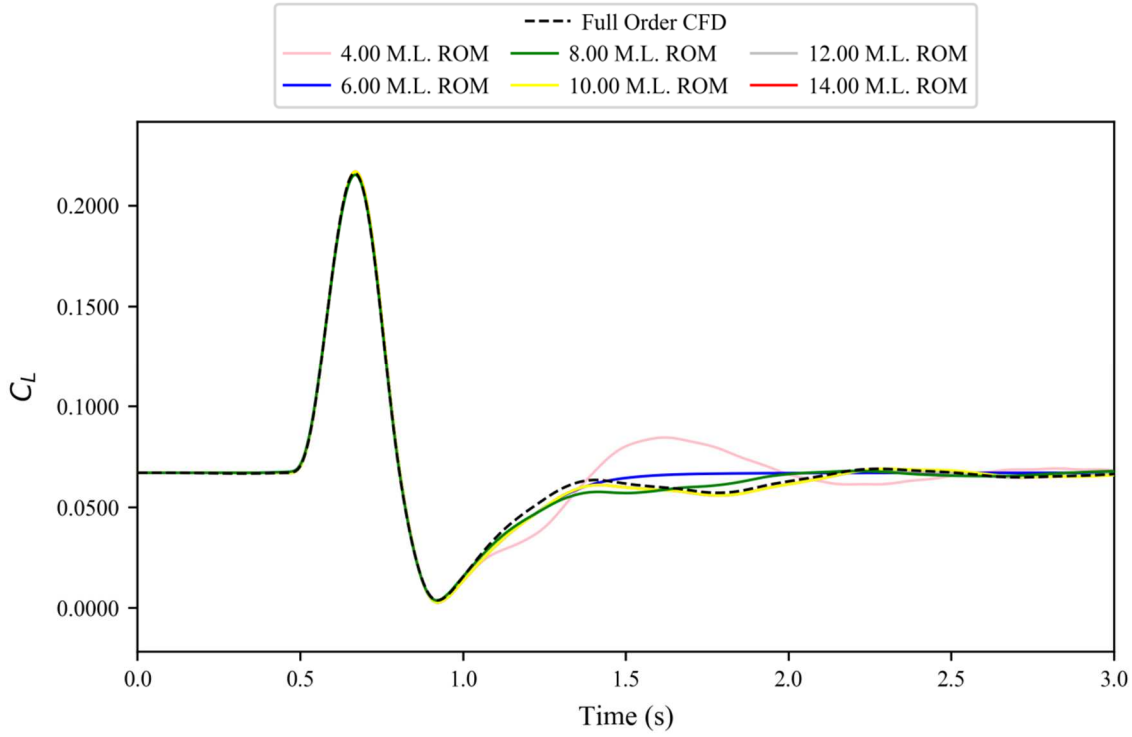

Figure 6.7. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of lift for a flight mechanics enabled model at flight point 2, gust case 2.

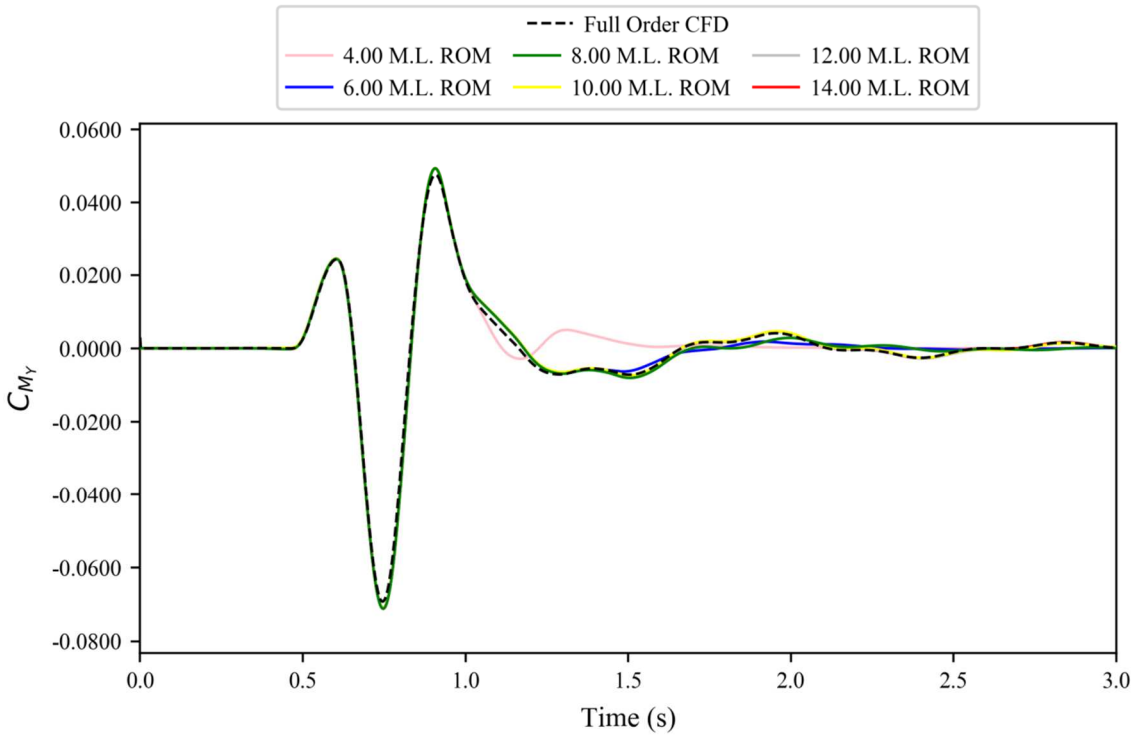

Figure 6.8. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 2.

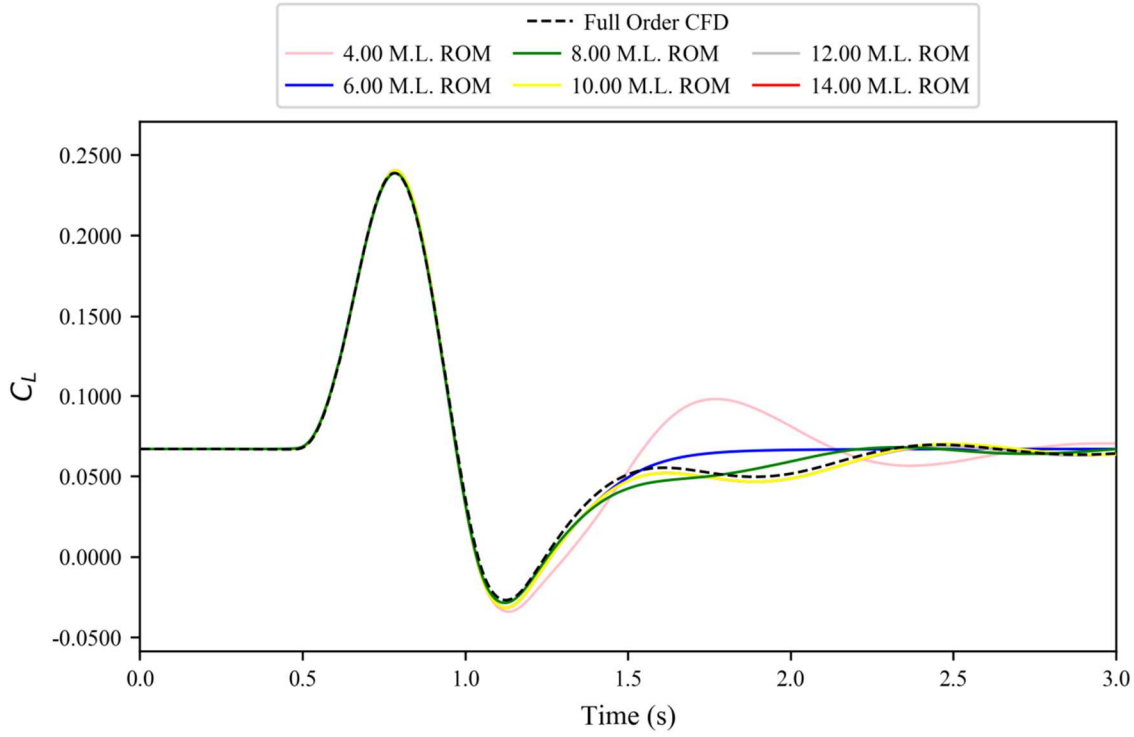

Figure 6.9. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 3.

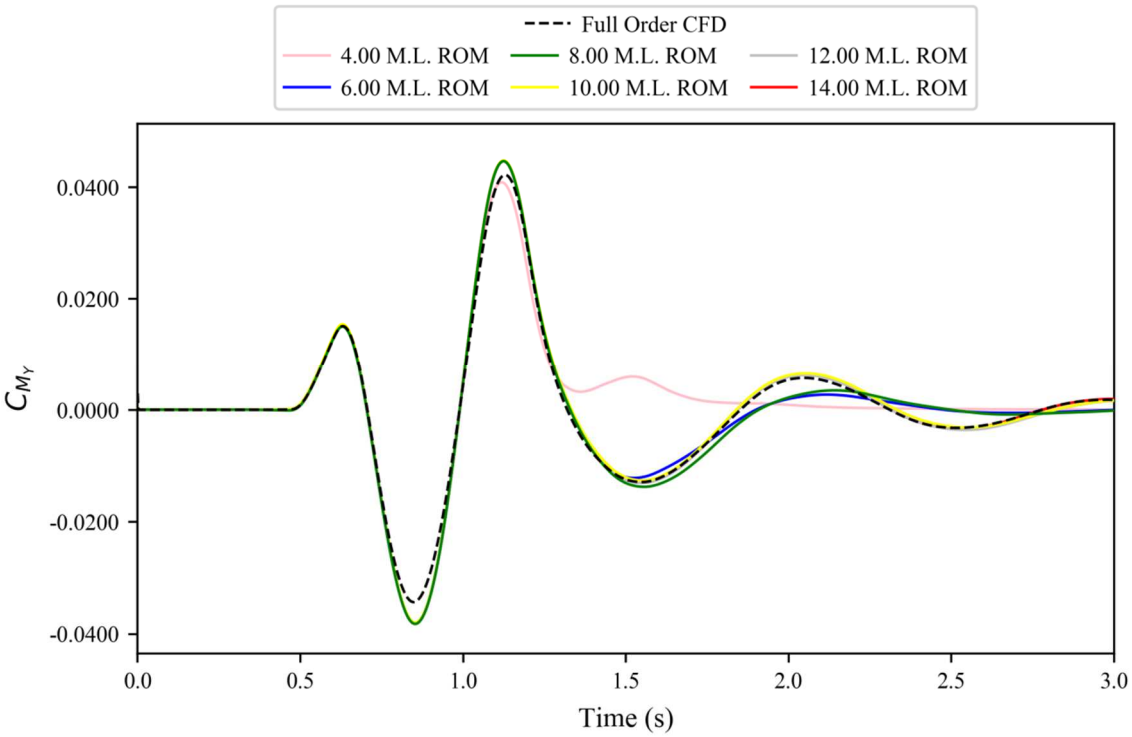

Figure 6.10. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 3.

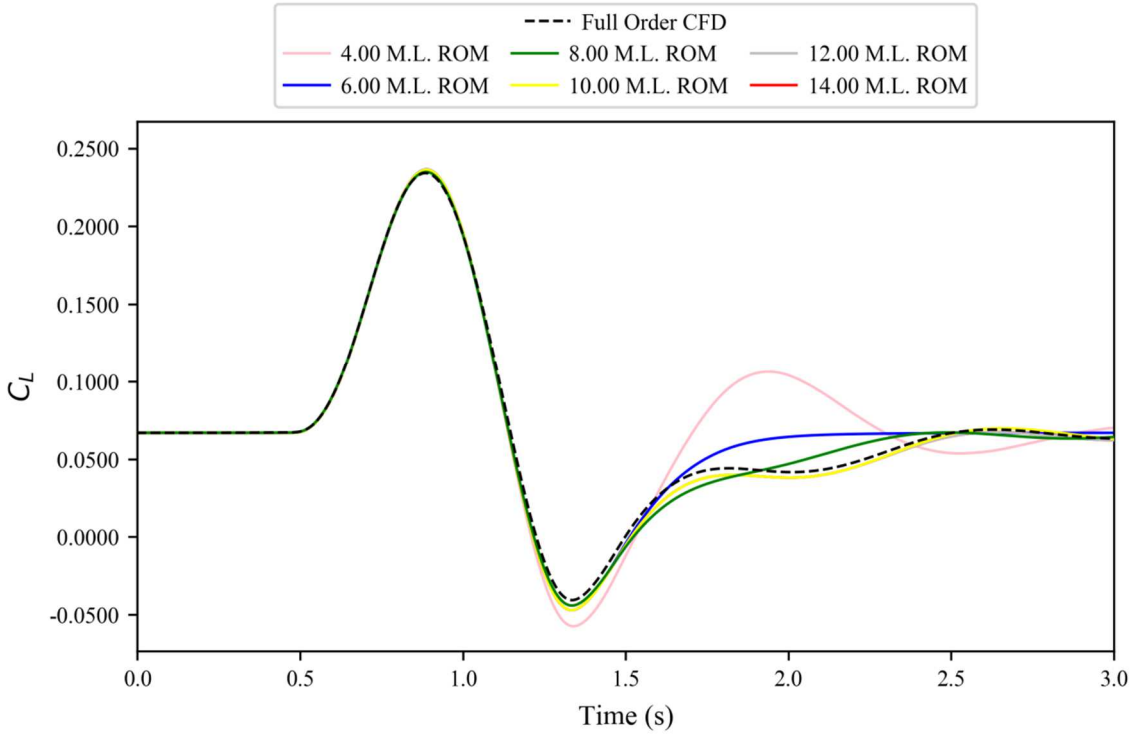

Figure 6.11. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of lift for a flight mechanics enabled model at flight point 2, gust case 4.

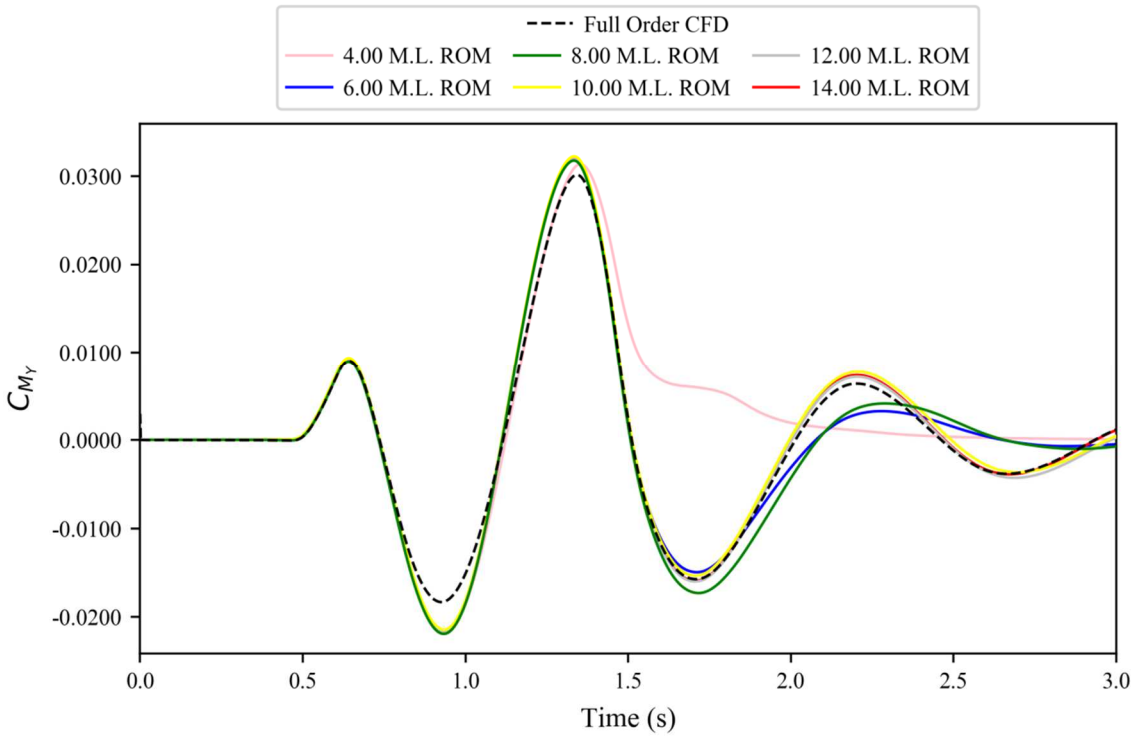

Figure 6.12. Various sharp-edged gust length based ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 4.

6.2. ROM Size

At the start of this section it was noted that the ROM size had been increased from 20, as used in the previous test cases, to 60 in order to capture more details for the more complex flight mechanics model. However, it is worth exploring the effects of ROM size in more detail.

To do this, ROMs were created with varying sizes, and their results compared to see at what point they appear to converge on a solution.

6.2.1. Results

Again, the new ROM method is tested on the gust cases of flight point 2, Figures 6.13-6.20 show that the ROM size of 60 was a reasonable choice. Even the smallest ROM, of size 15, captures the behaviour of the system well; however, the results do not appear to fully converge until a ROM size of 60. Therefore it is logical to take this size forward as it maximises accuracy whilst also keeping the computational costs as low as possible;

although it should be noted that in this instance the computational savings will mostly be applicable in cases where a ROM is created and then used repeatedly.

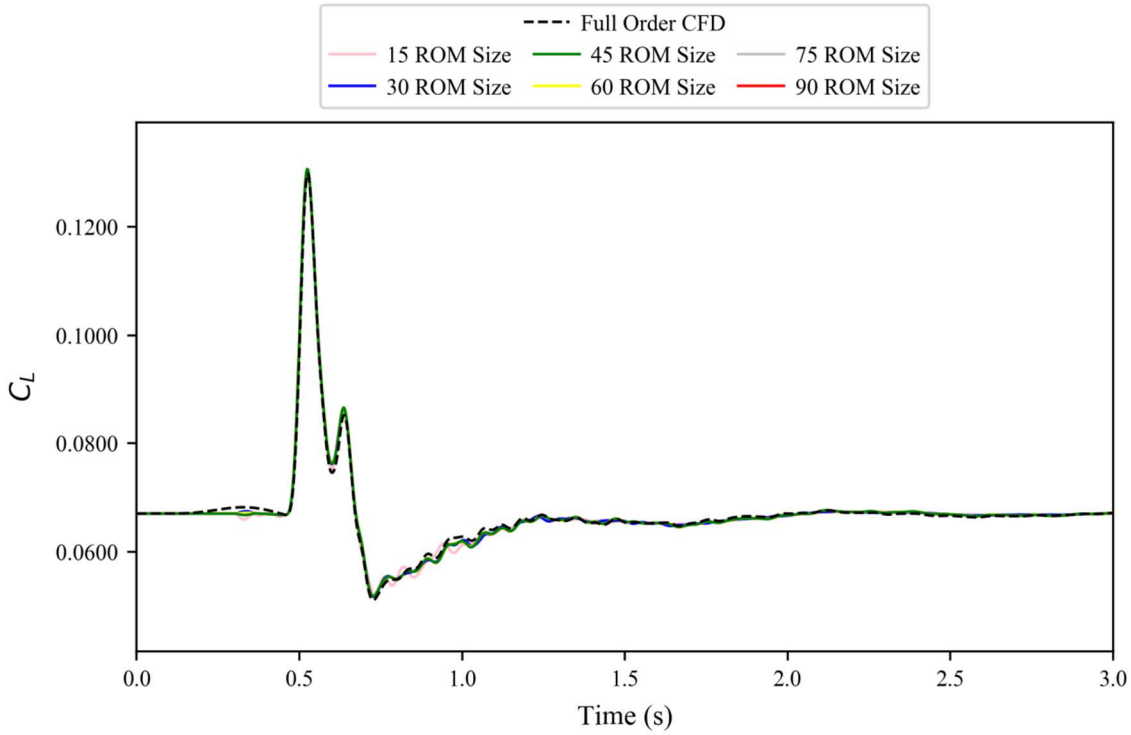

Figure 6.13. Various sized ROM results against full order CFD simulation for the coefficient of lift for a flight mechanics enabled model at flight point 2, gust case 1.

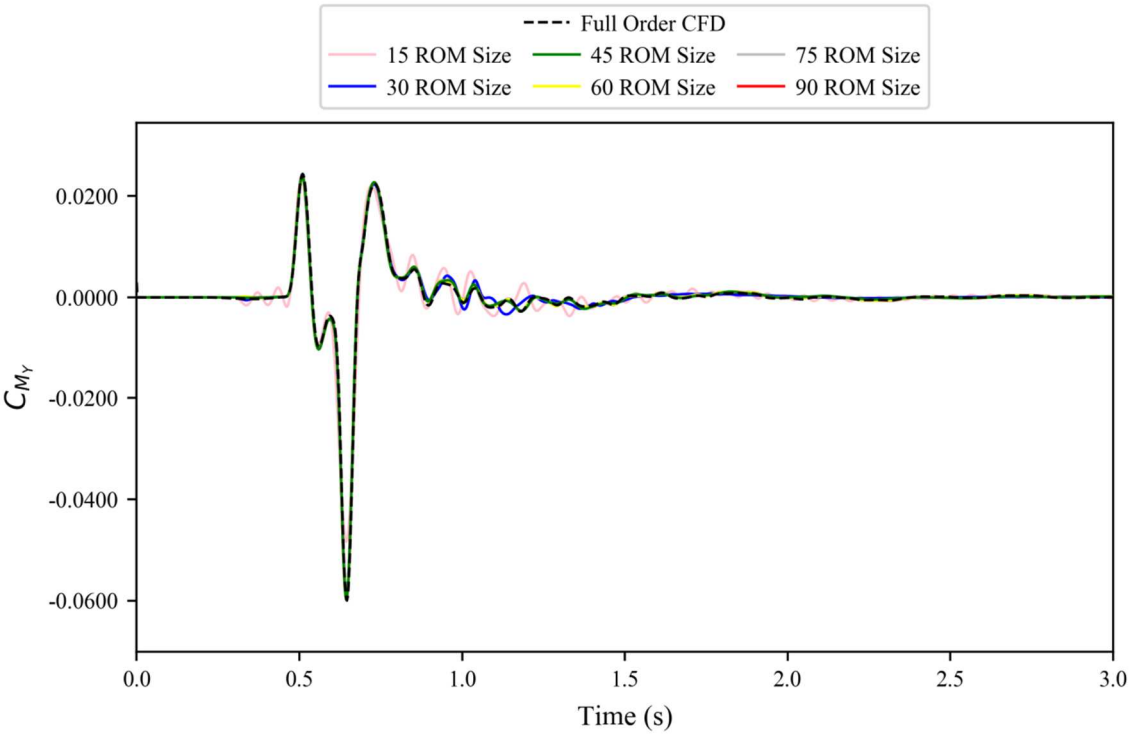

Figure 6.14. Various sized ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 1.

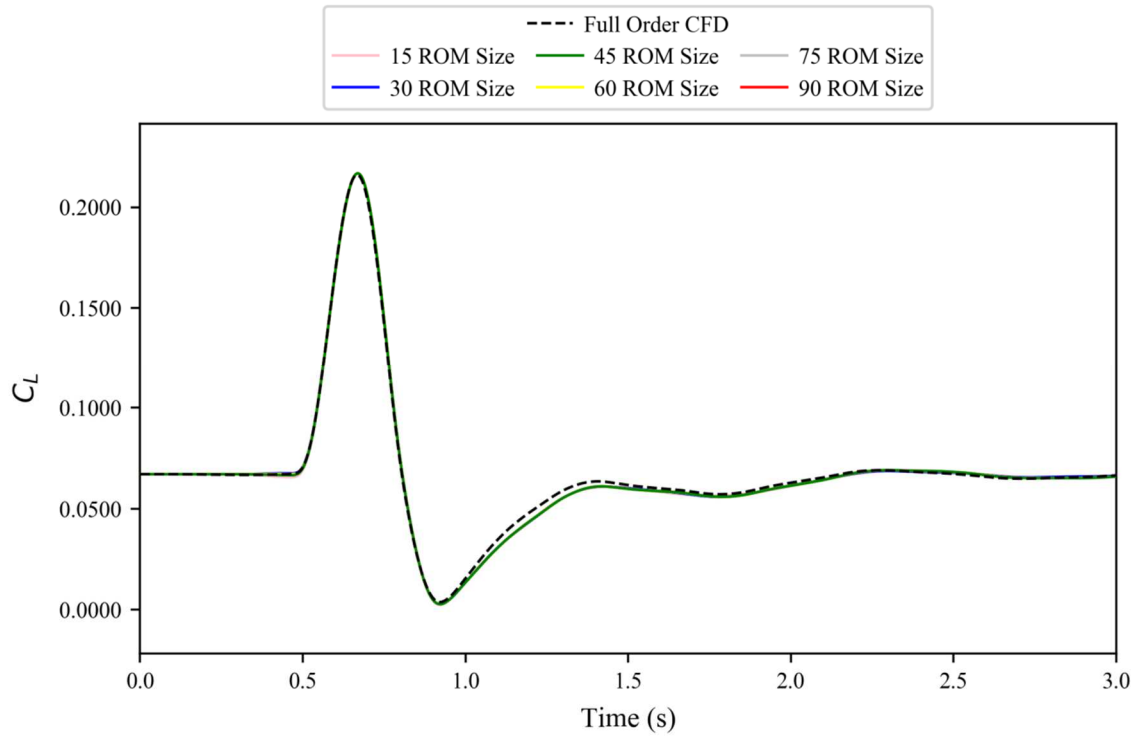

Figure 6.15. Various sized ROM results against full order CFD simulation for the coefficient of lift for a flight mechanics enabled model at flight point 2, gust case 2.

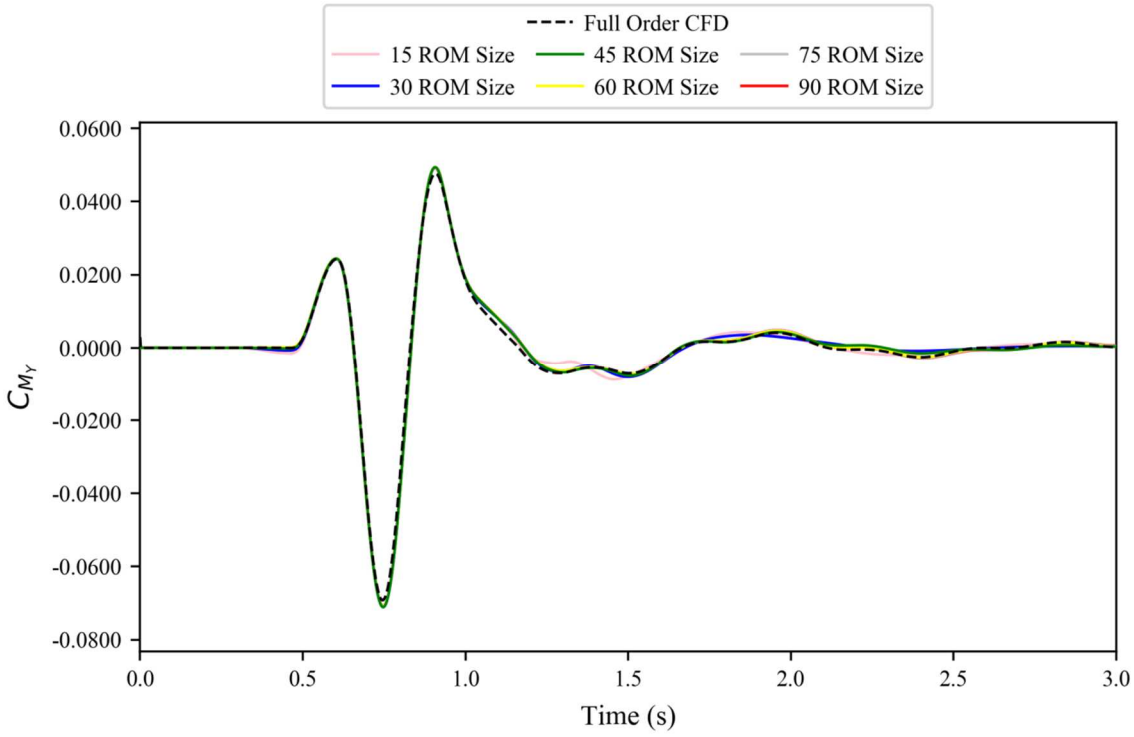

Figure 6.16. Various sized ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 2.

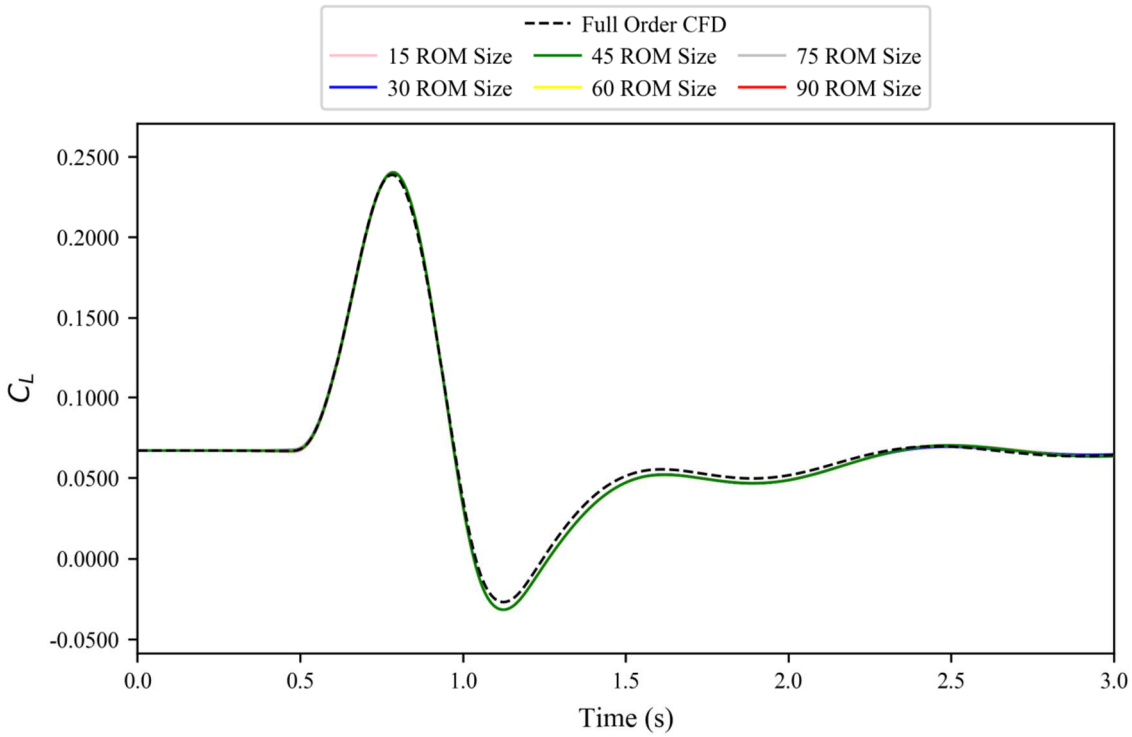

Figure 6.17. Various sized ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 3.

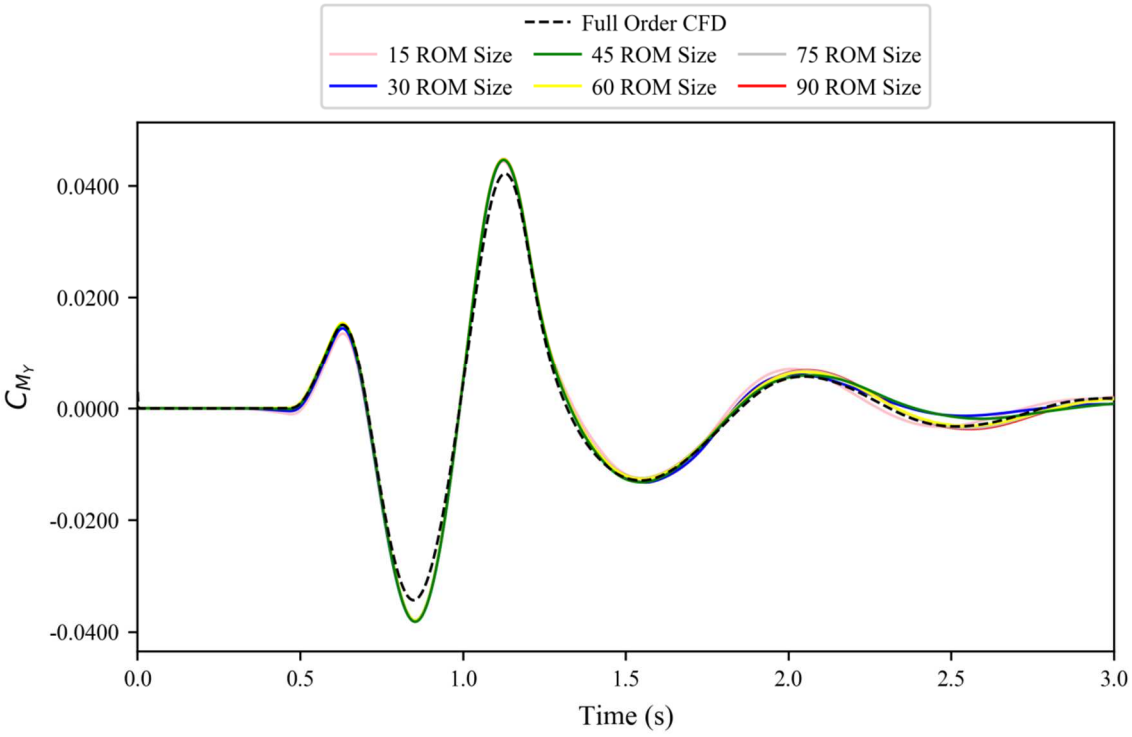

Figure 6.18. Various sized based ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 3.

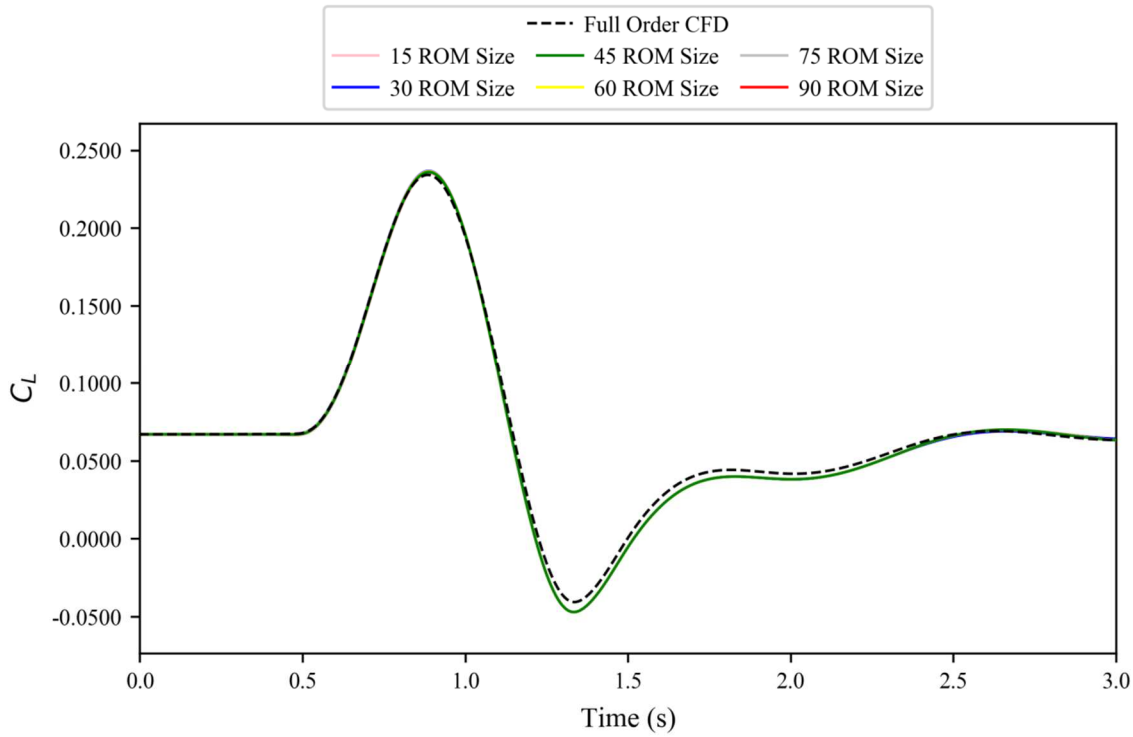

Figure 6.19. Various sized ROM results against full order CFD simulation for the coefficient of lift for a flight mechanics enabled model at flight point 2, gust case 4.

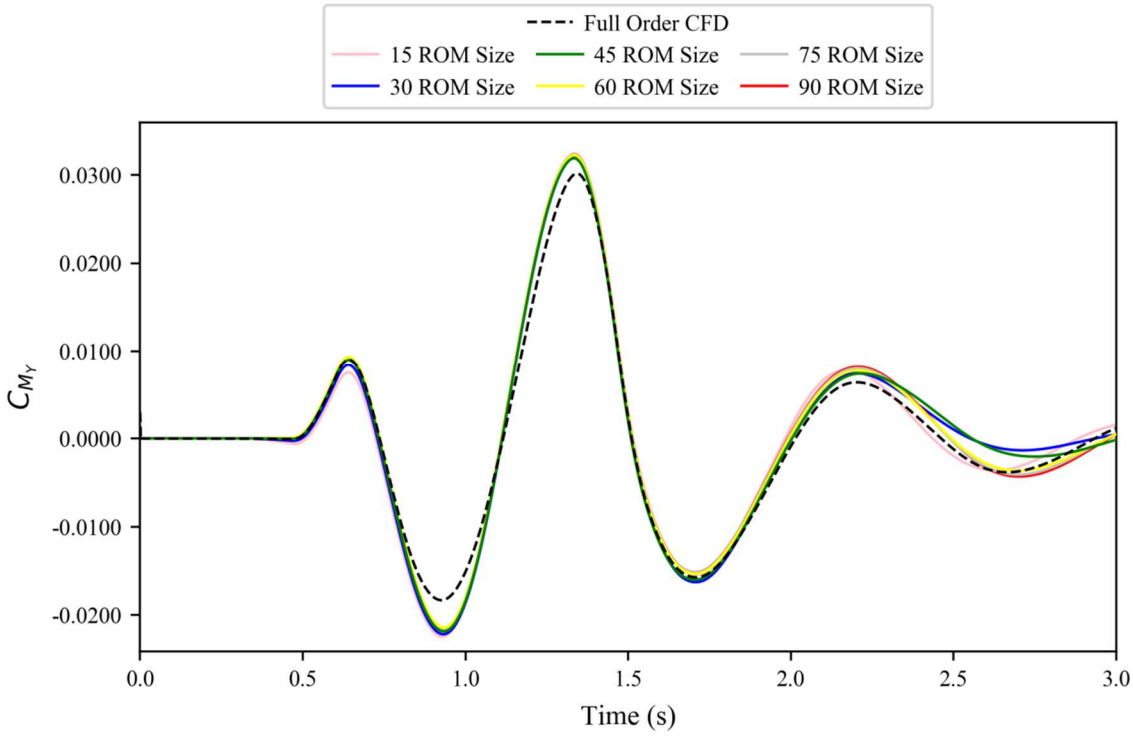

Figure 6.20. Various sized ROM results against full order CFD simulation for the coefficient of pitching moment for a flight mechanics enabled model at flight point 2, gust case 4.

6.3. Additional Results

6.3.1. Overview

The final ROM method presented in this chapter is built using 9 (post-impact) model lengths worth of data in the sharp-edged gust, and with a ROM size of 60. As before, whilst flight point 2 has already been looked at in detail (see Figures 6.13-6.20) it is worth exploring the other four gust points laid out in Table 6 in order to gain a more complete understanding on how well the ROM performs.

The results (Figures 6.13-6.20 and 6.21-6.52) show an interesting pattern. For the shorter gusts, the results are very strong; nearly identical to the full order CFD results. However, as the gust length increases the accuracy rapidly diminishes; at a much greater rate than previously seen. It can also be seen that this degradation appears to get dramatically worse for flight point 5.

6.3.2. Flight Point 1

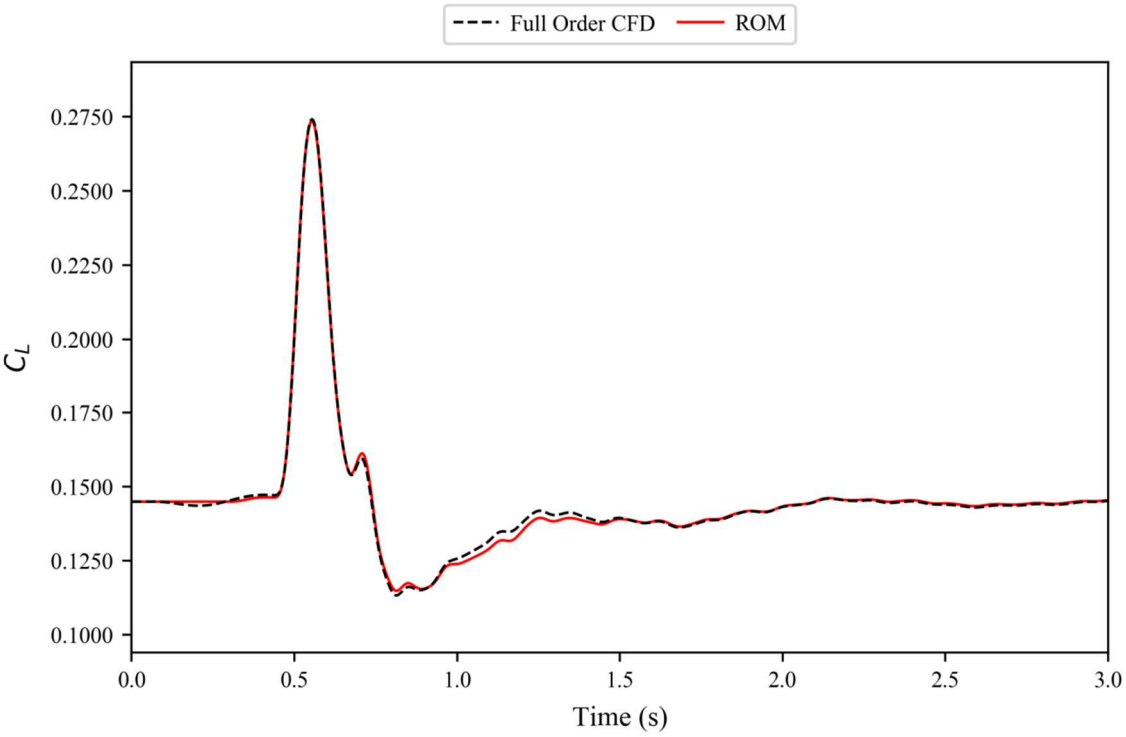

Figure 6.21. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 1, gust case 1.

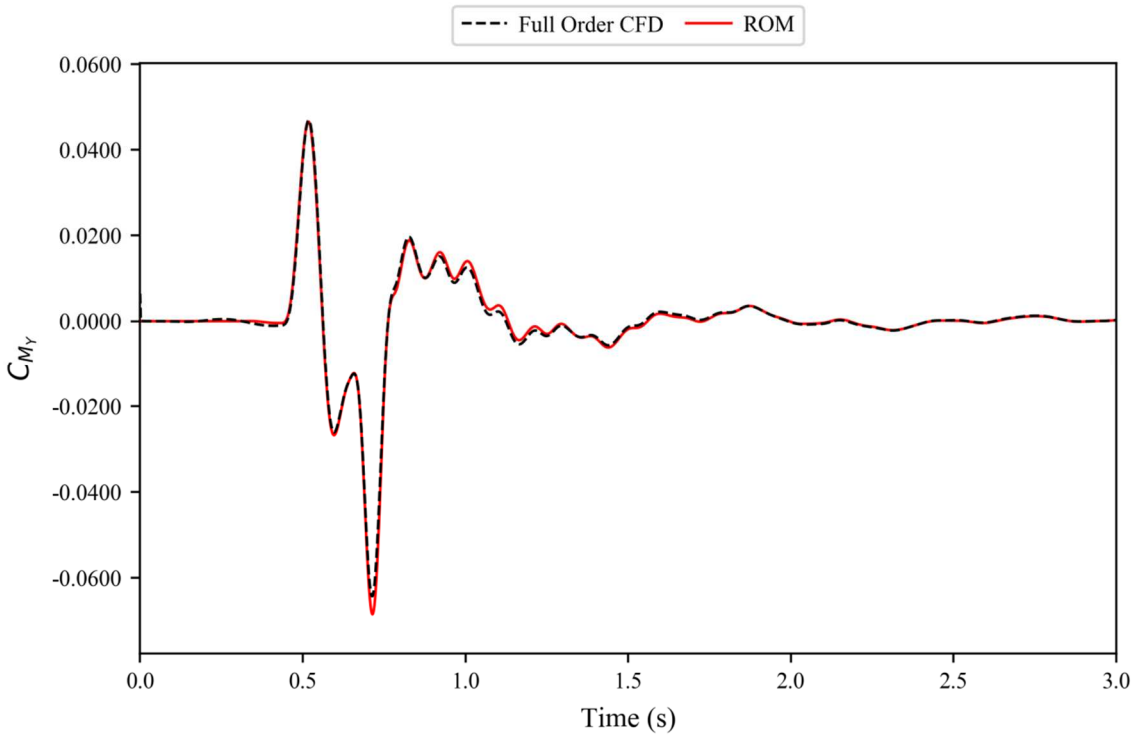

Figure 6.22. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 1, gust case 1.

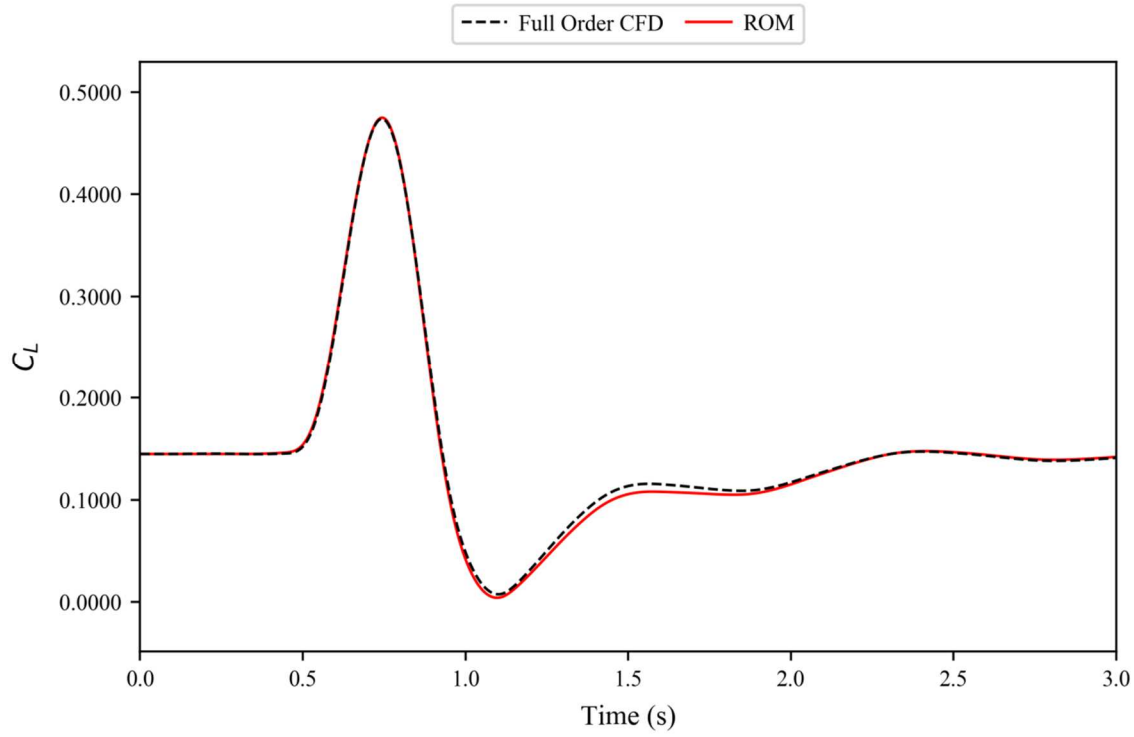

Figure 6.23. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 1, gust case 2.

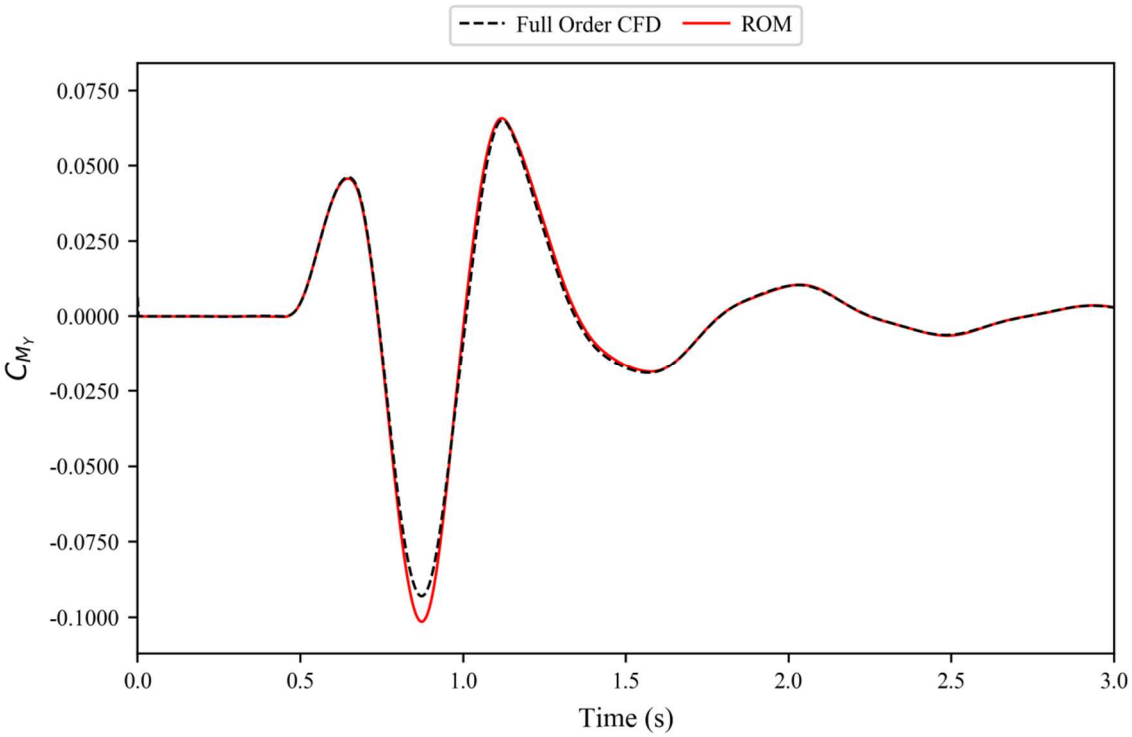

Figure 6.24. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 1, gust case 2.

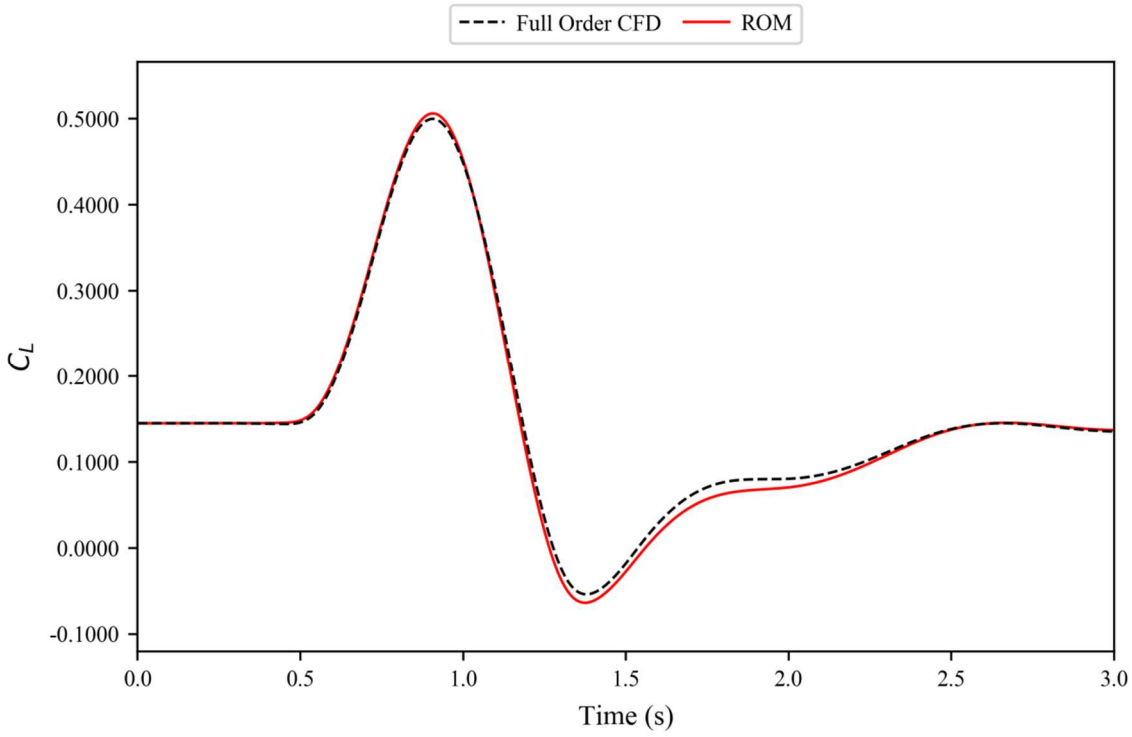

Figure 6.25. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 1, gust case 3.

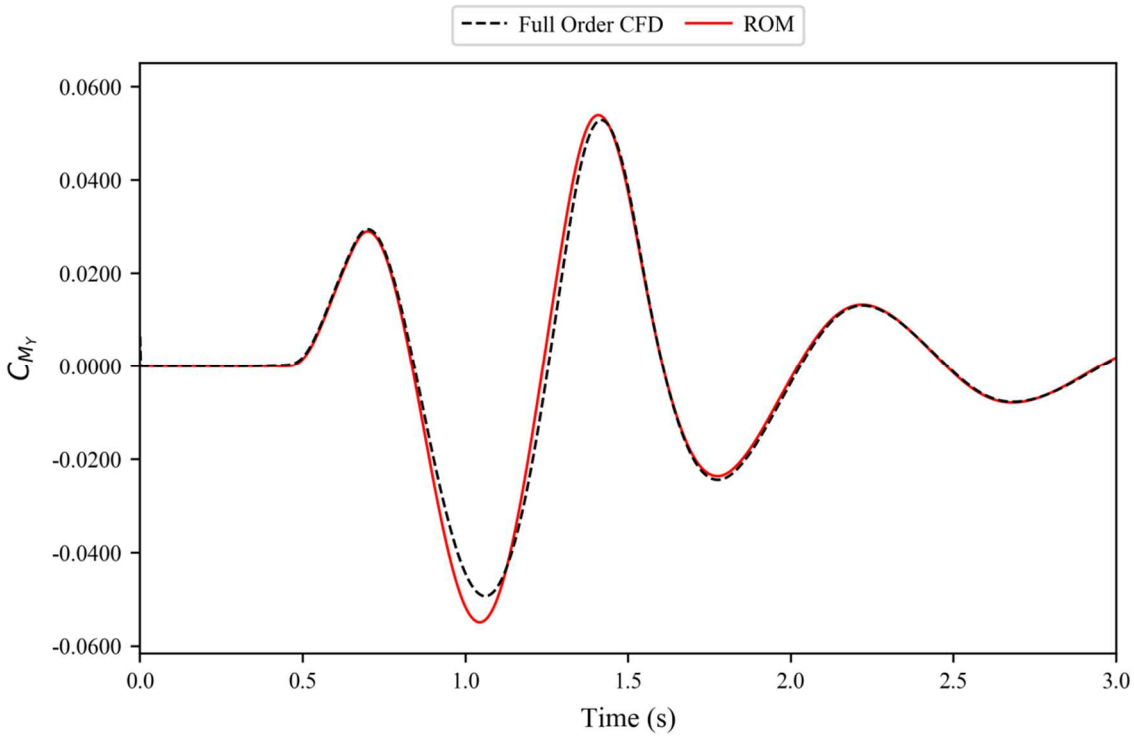

Figure 6.26. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 1, gust case 3.

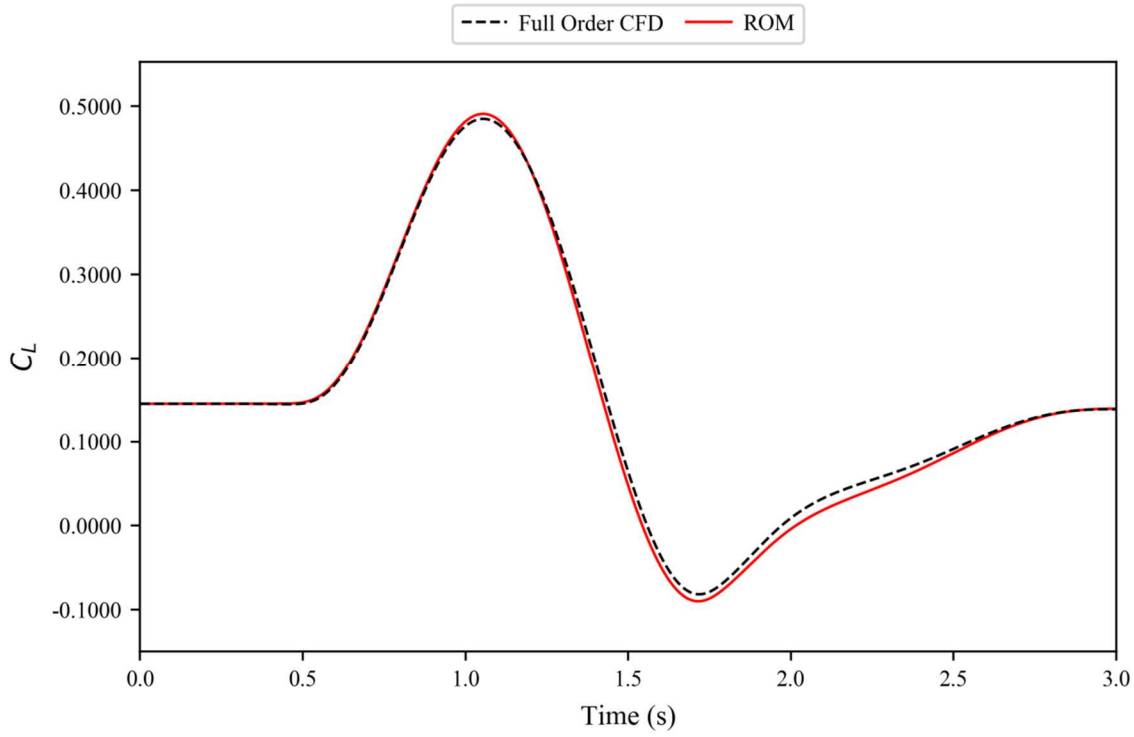

Figure 6.27. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 1, gust case 4.

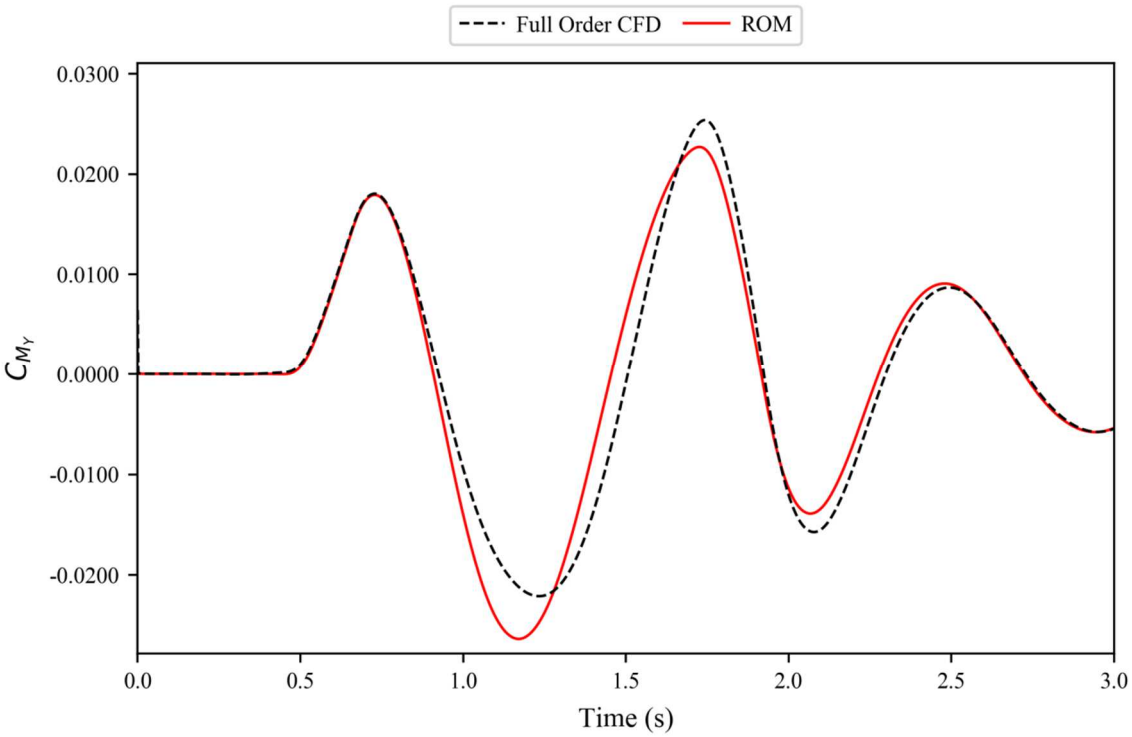

Figure 6.28. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 1, gust case 4.

6.3.3. Flight Point 3

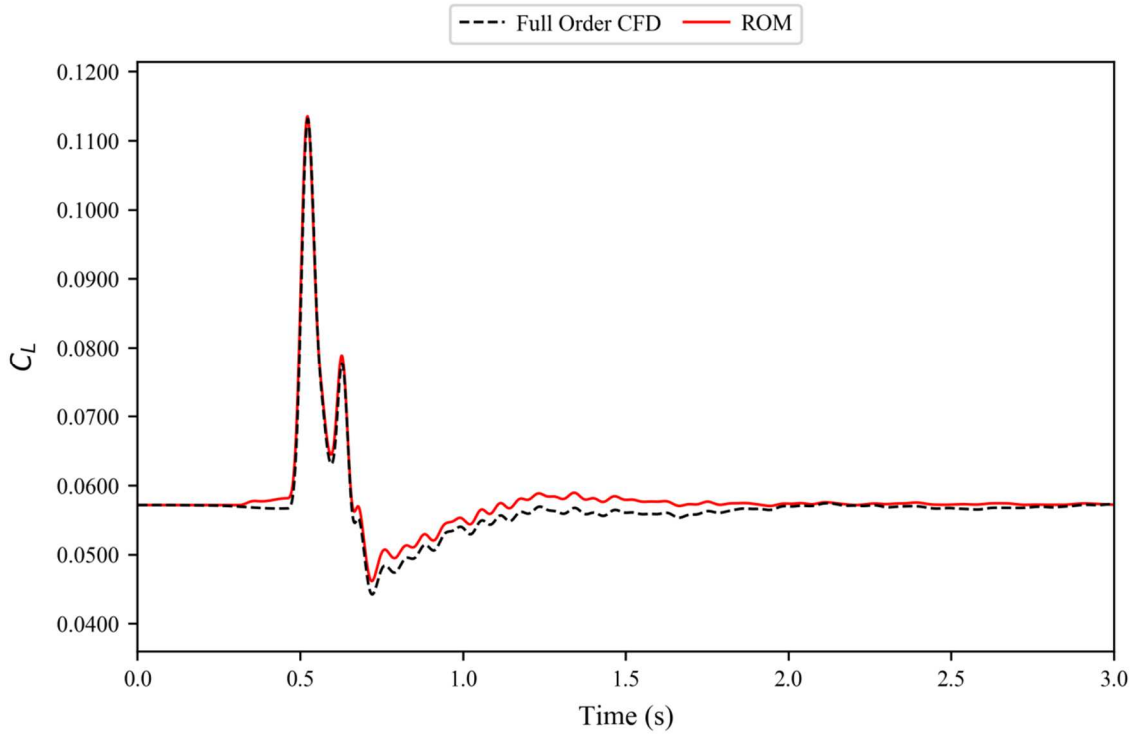

Figure 6.29. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 3, gust case 1.

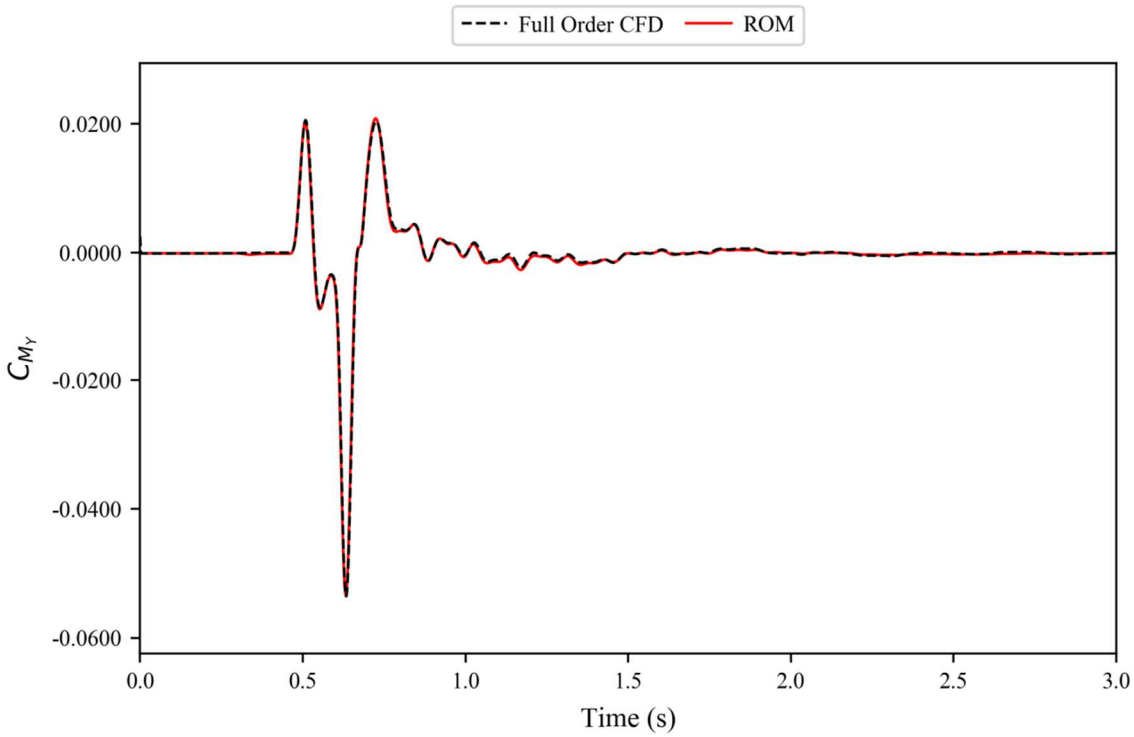

Figure 6.30. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 3, gust case 1.

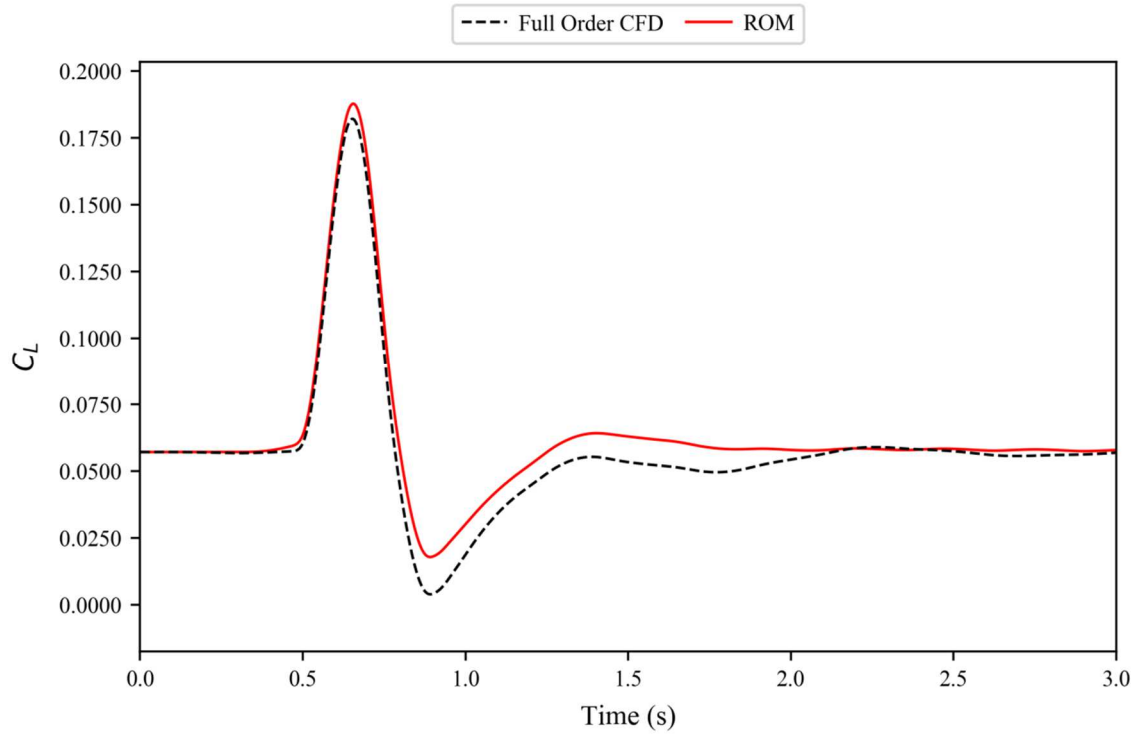

Figure 6.31. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 3, gust case 2.

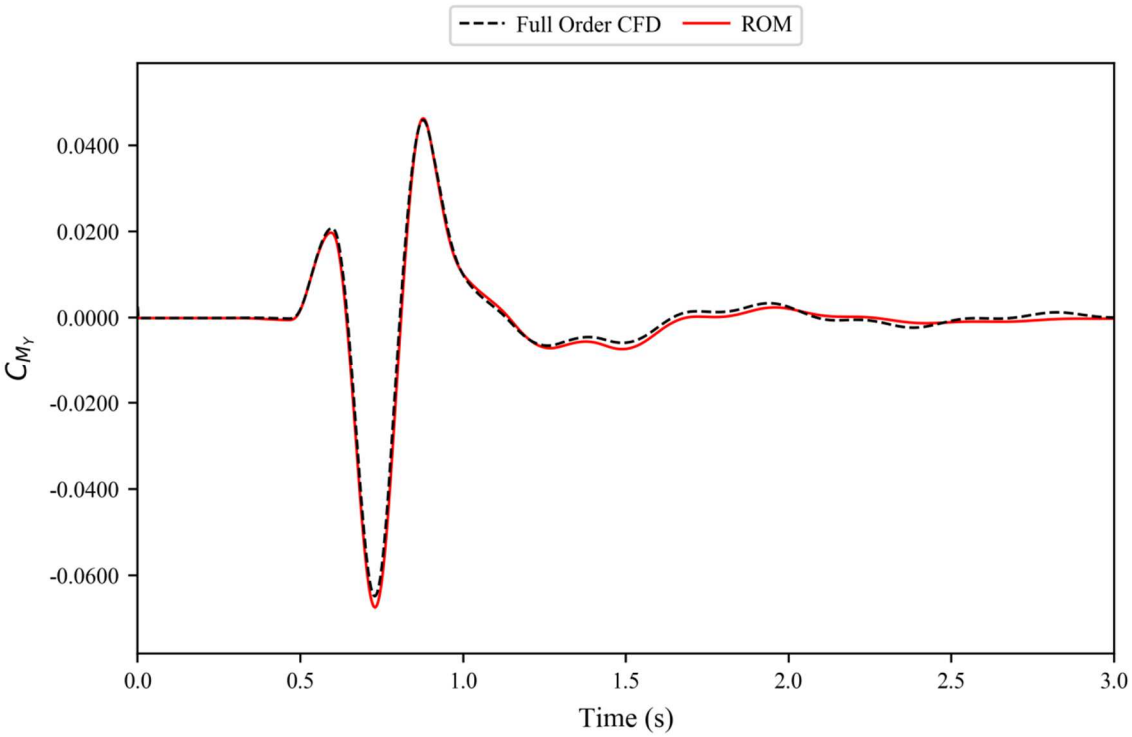

Figure 6.32. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 3, gust case 2.

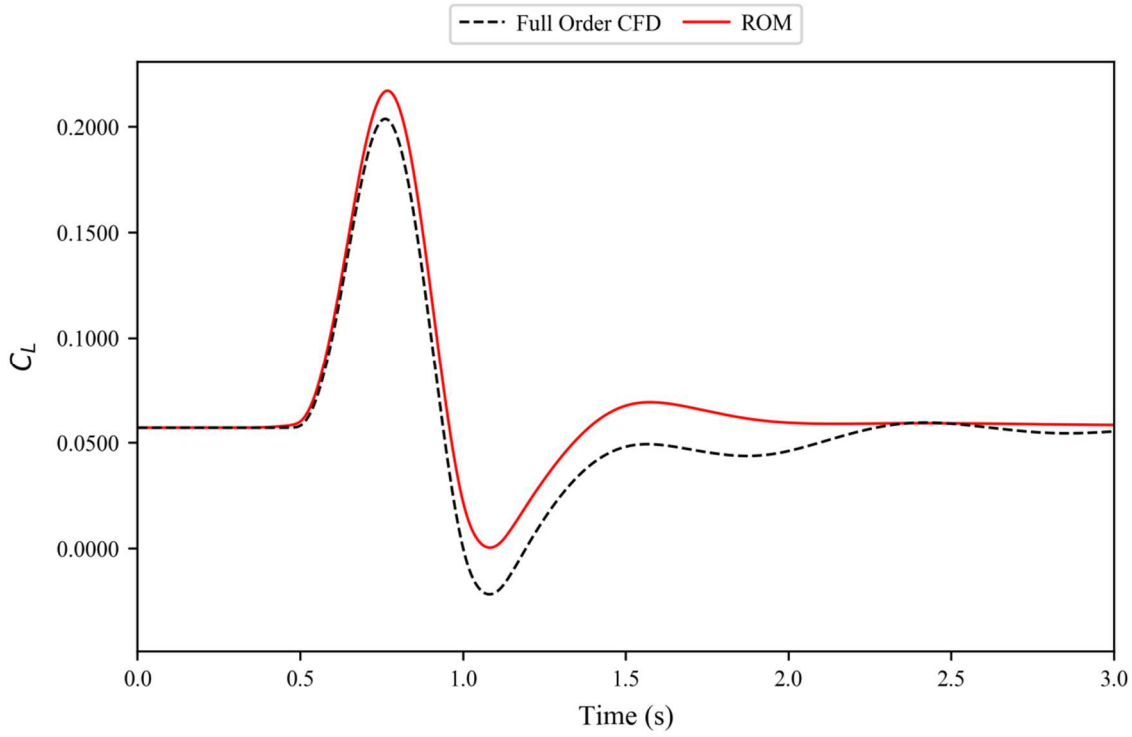

Figure 6.33. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 3, gust case 3.

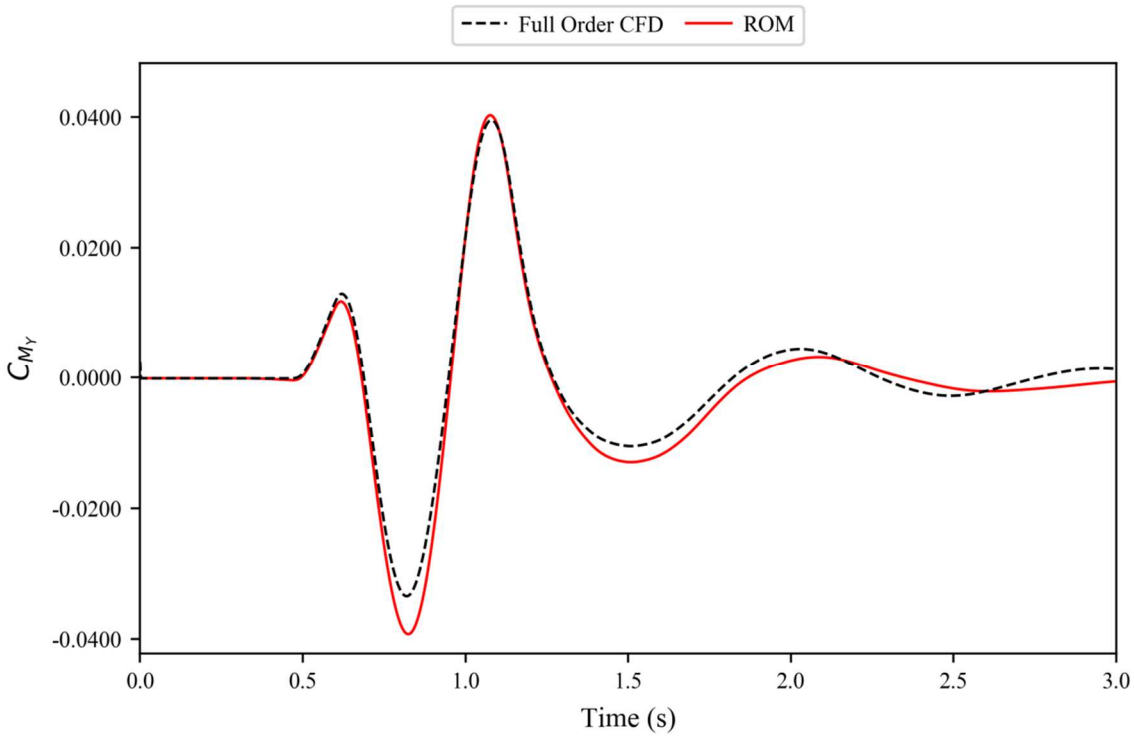

Figure 6.34. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 3, gust case 3.

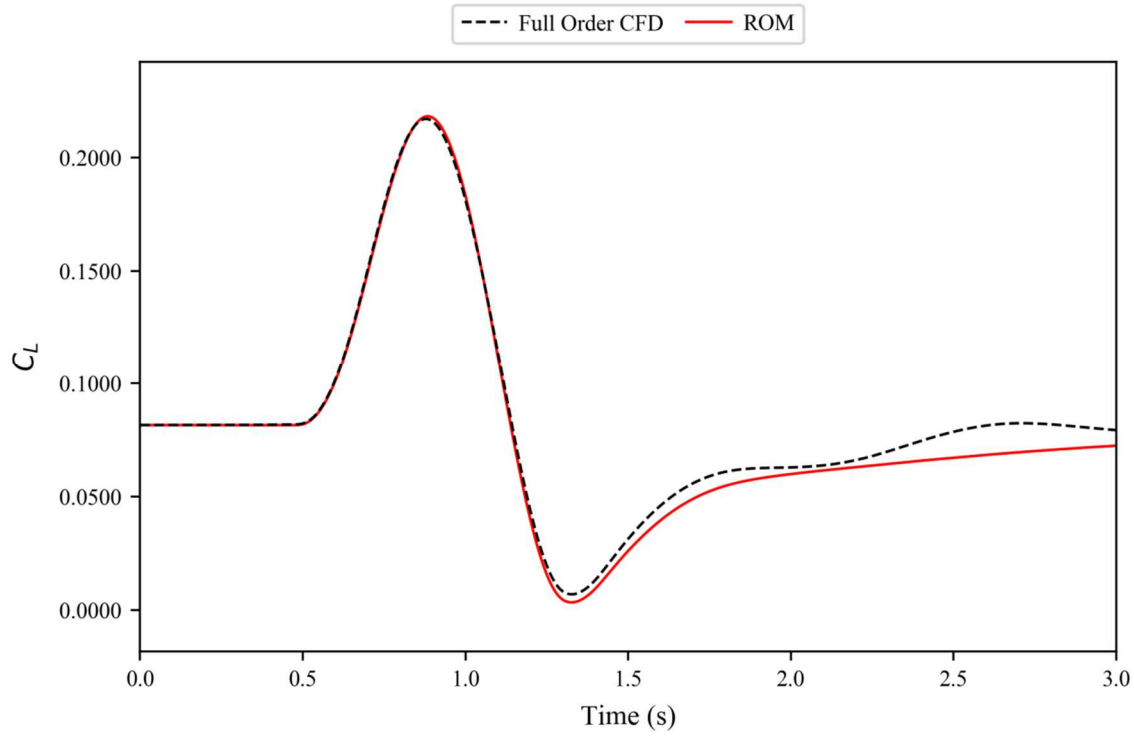

Figure 6.35. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 3, gust case 4.

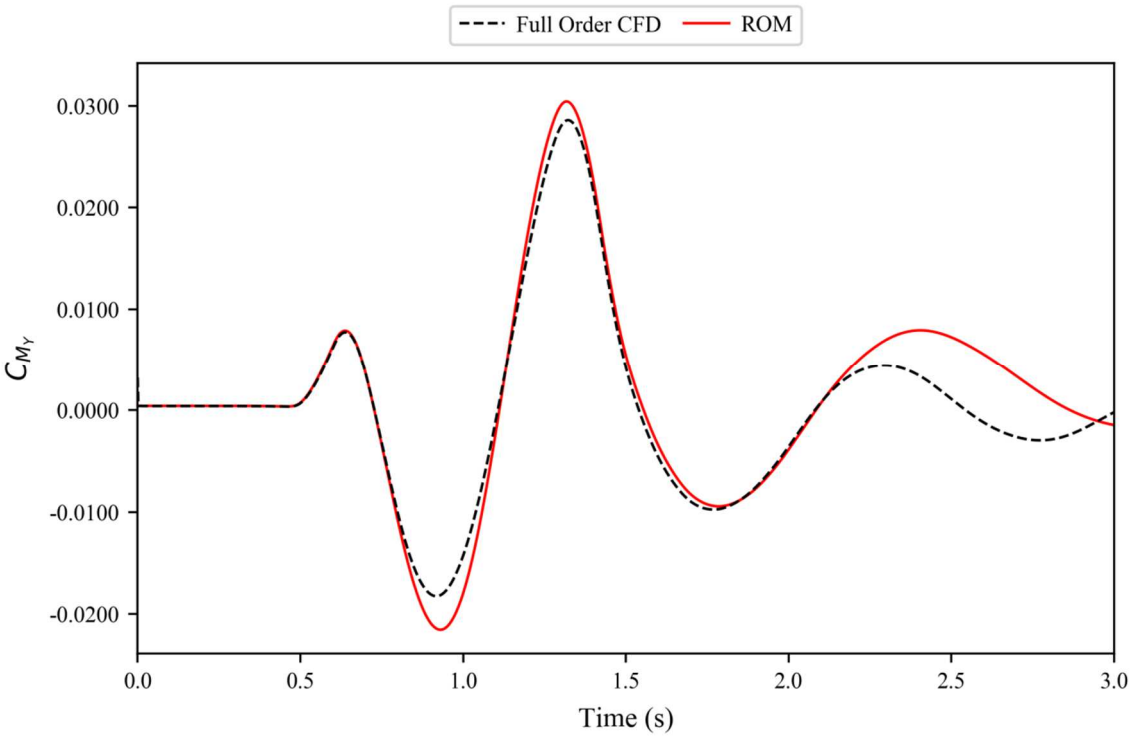

Figure 6.36. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 3, gust case 4.

6.3.4. Flight Point 4

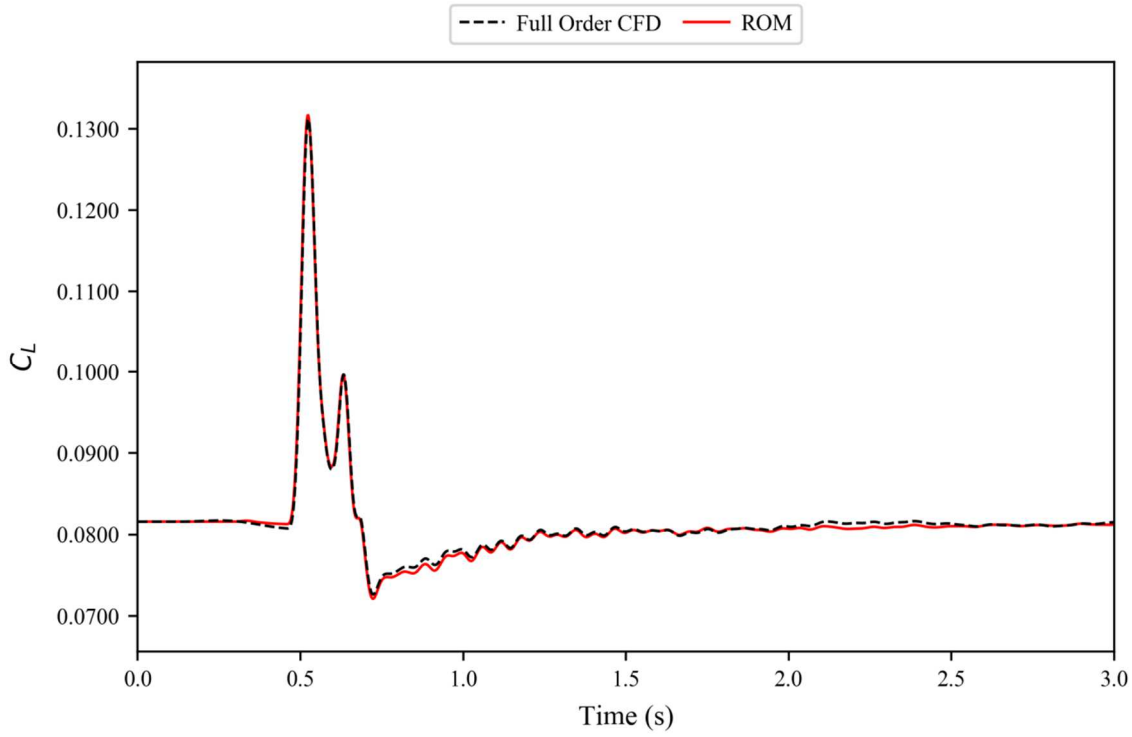

Figure 6.37. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 4, gust case 1.

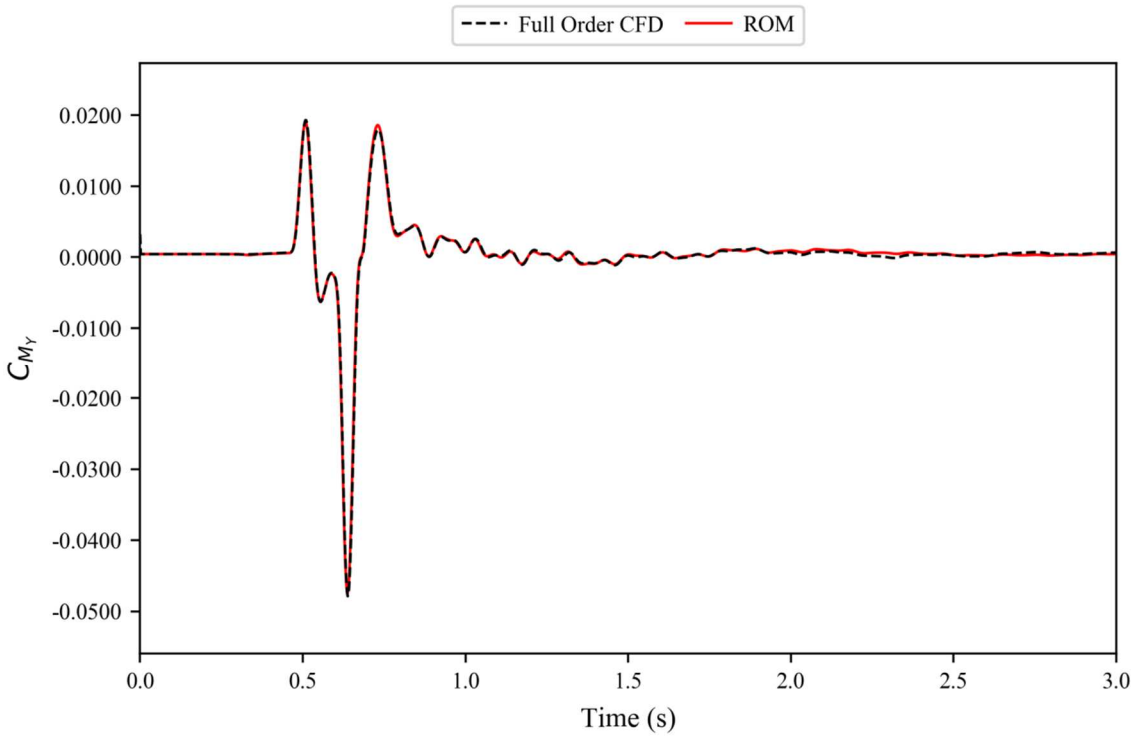

Figure 6.38. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 4, gust case 1.

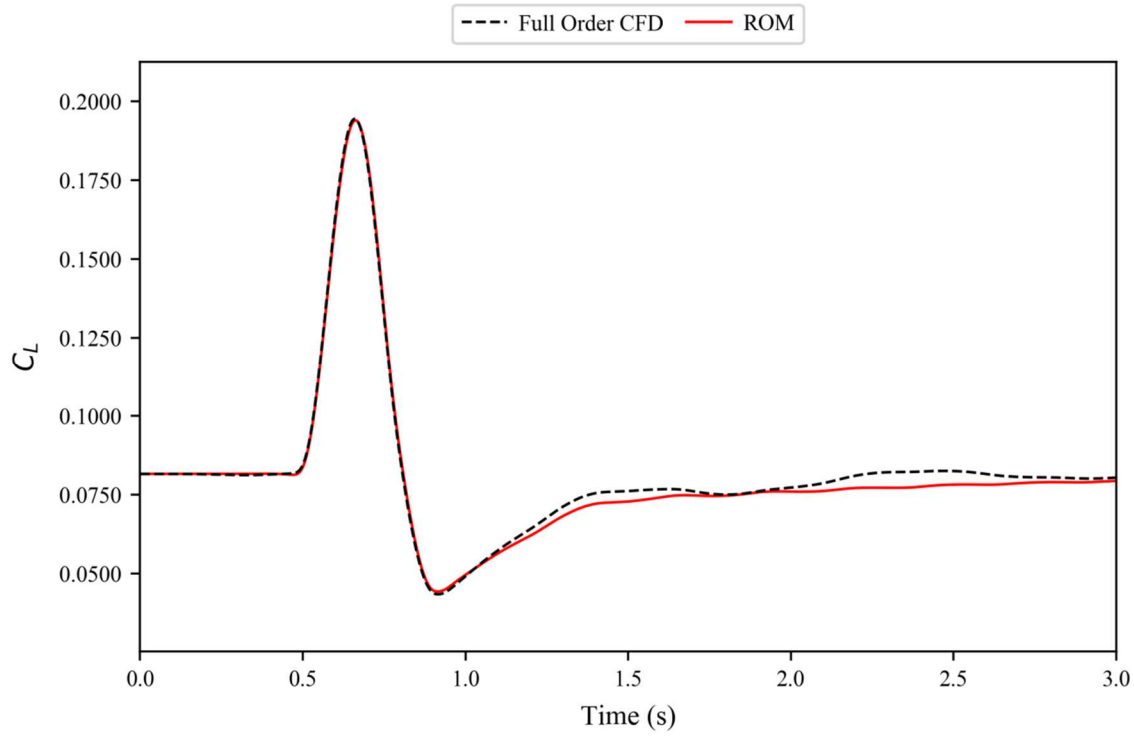

Figure 6.39. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 4, gust case 2.

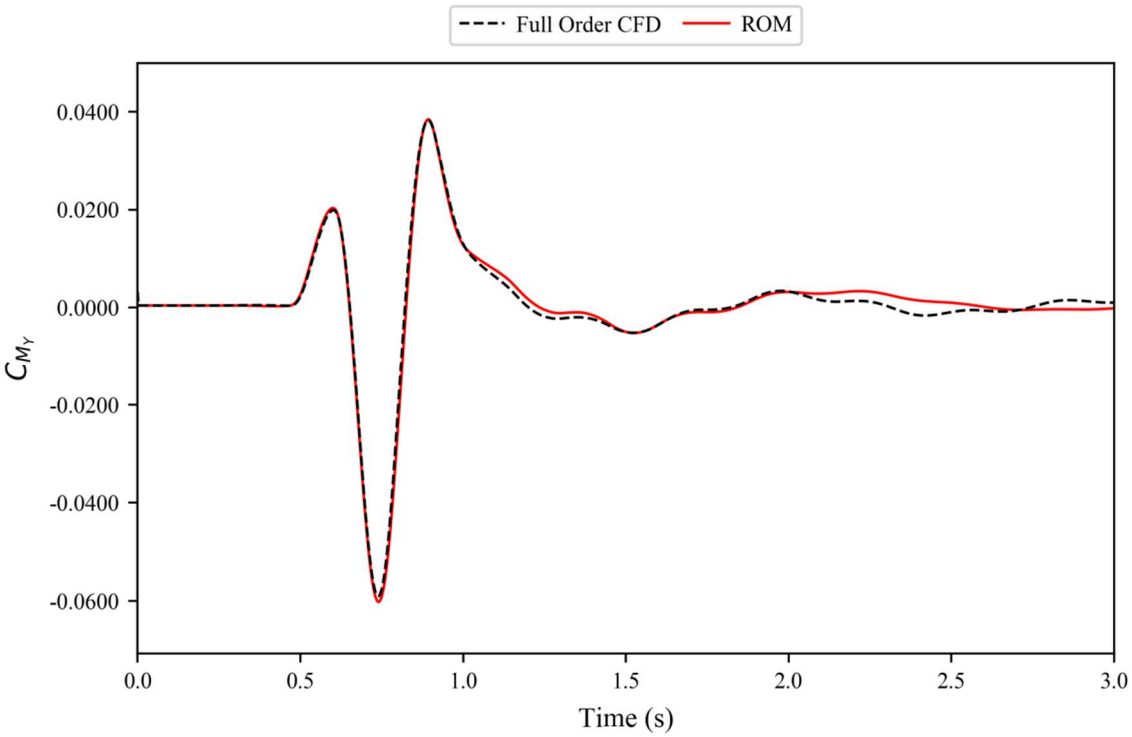

Figure 6.40. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 4, gust case 2.

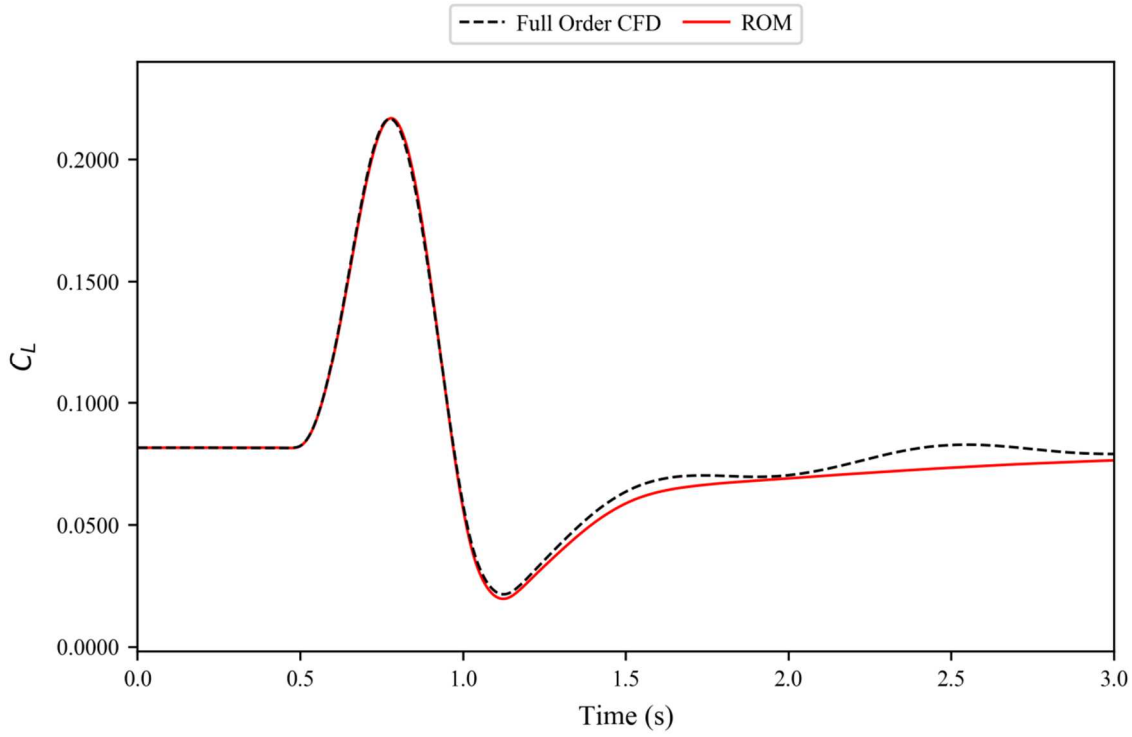

Figure 6.41. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 4, gust case 3.

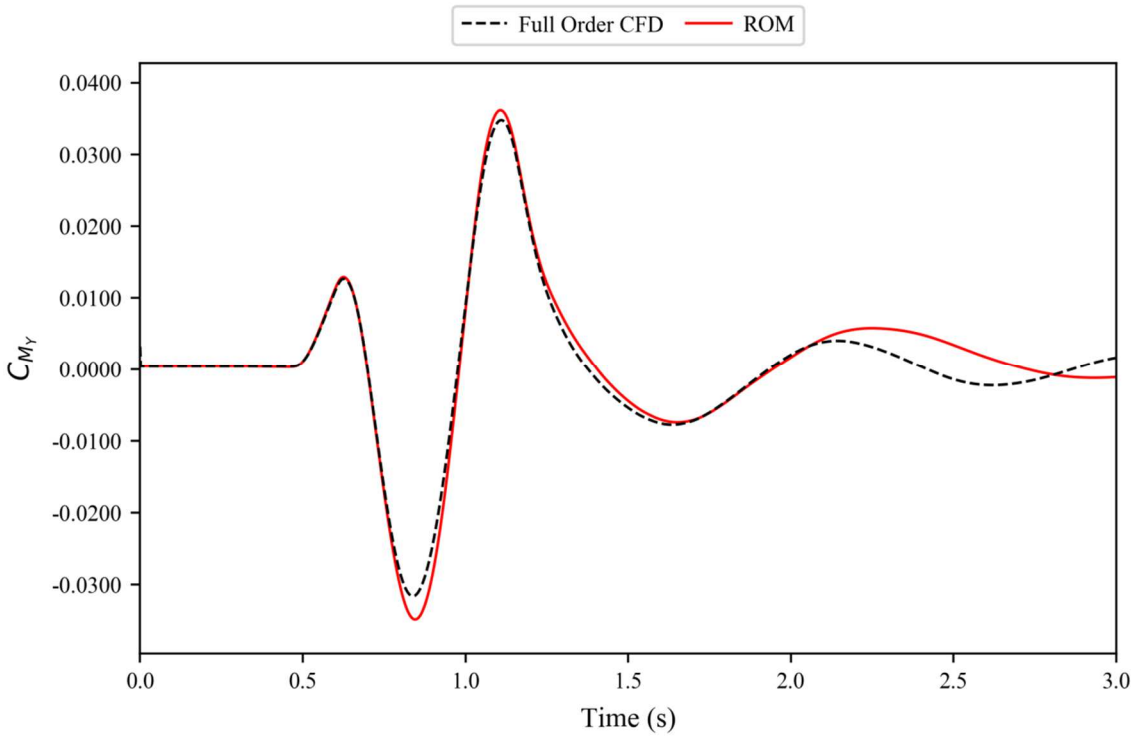

Figure 6.42. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 4, gust case 3.

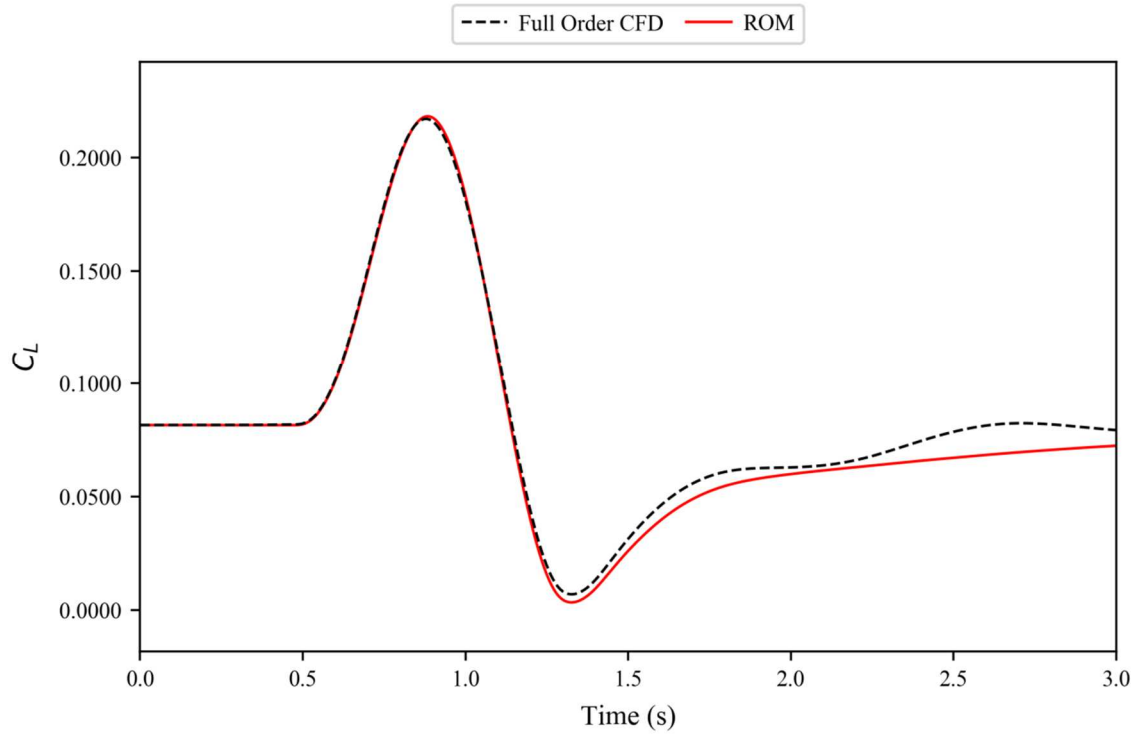

Figure 6.43. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 4, gust case 4.

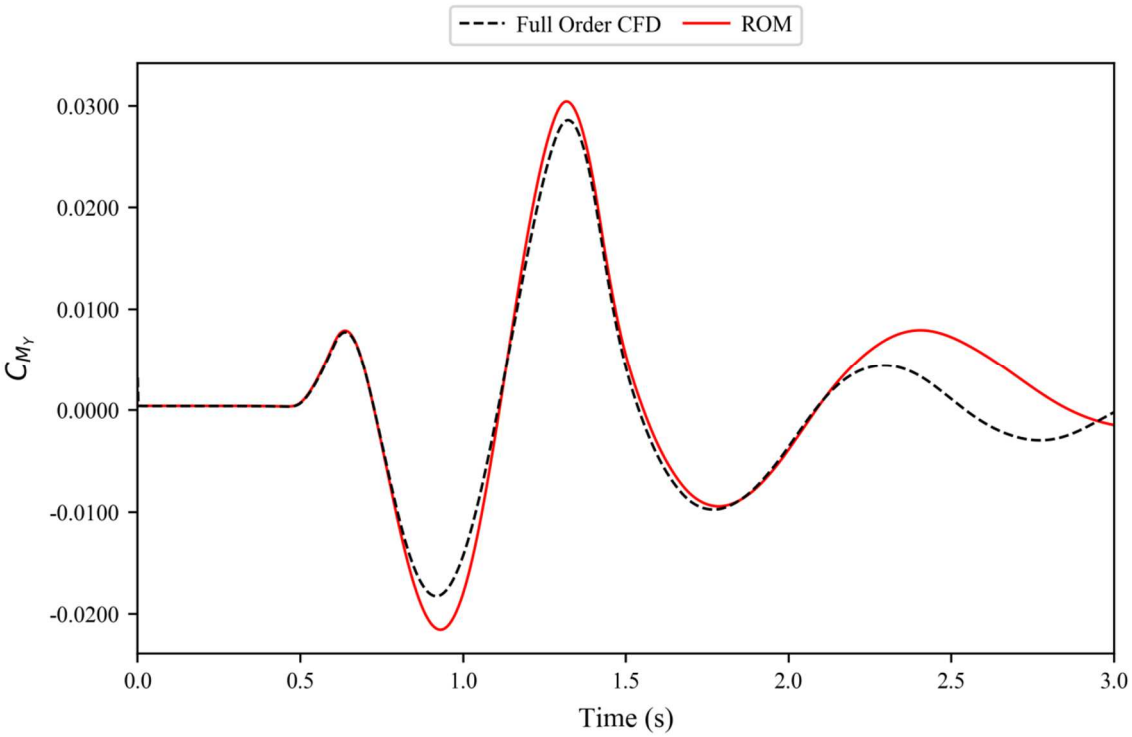

Figure 6.44. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 4, gust case 4.

6.3.5. Flight Point 5

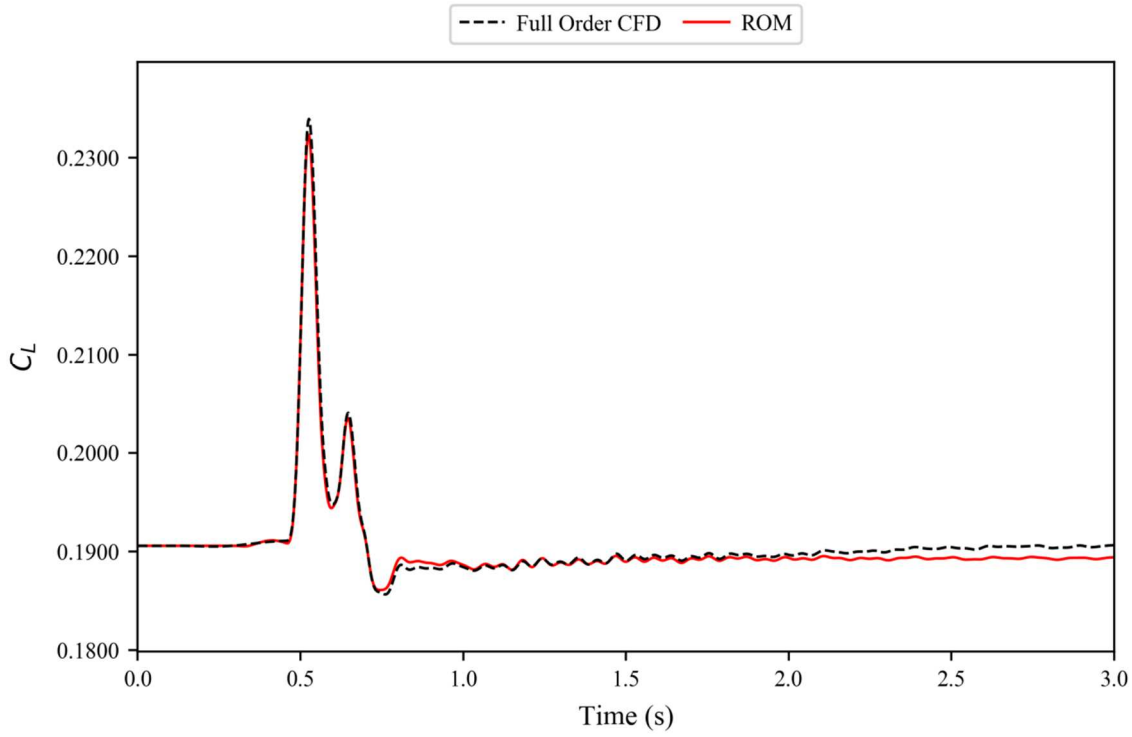

Figure 6.45. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 5, gust case 1.

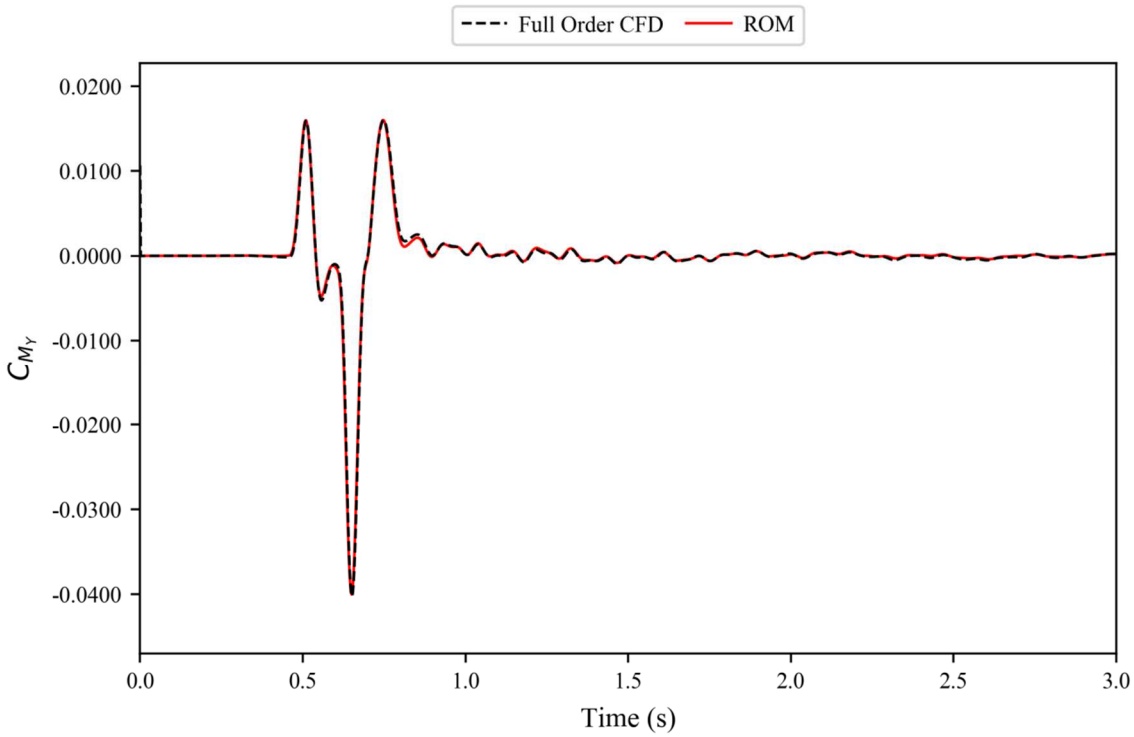

Figure 6.46. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 5, gust case 1.

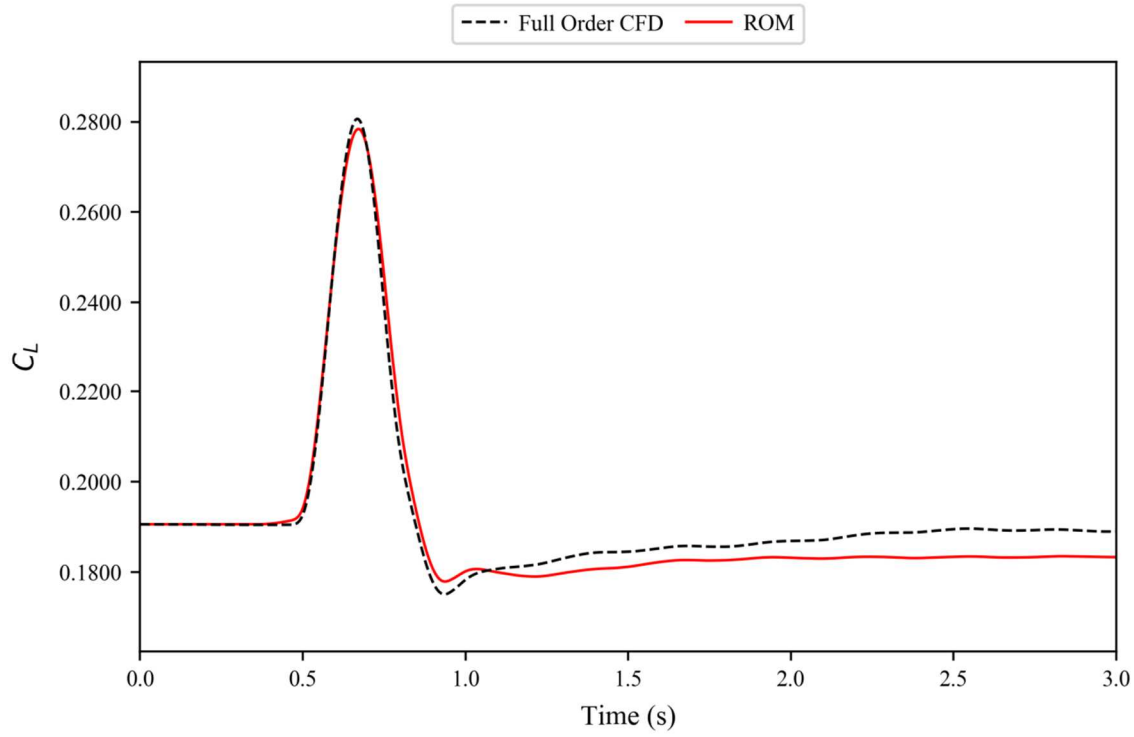

Figure 6.47. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 5, gust case 2.

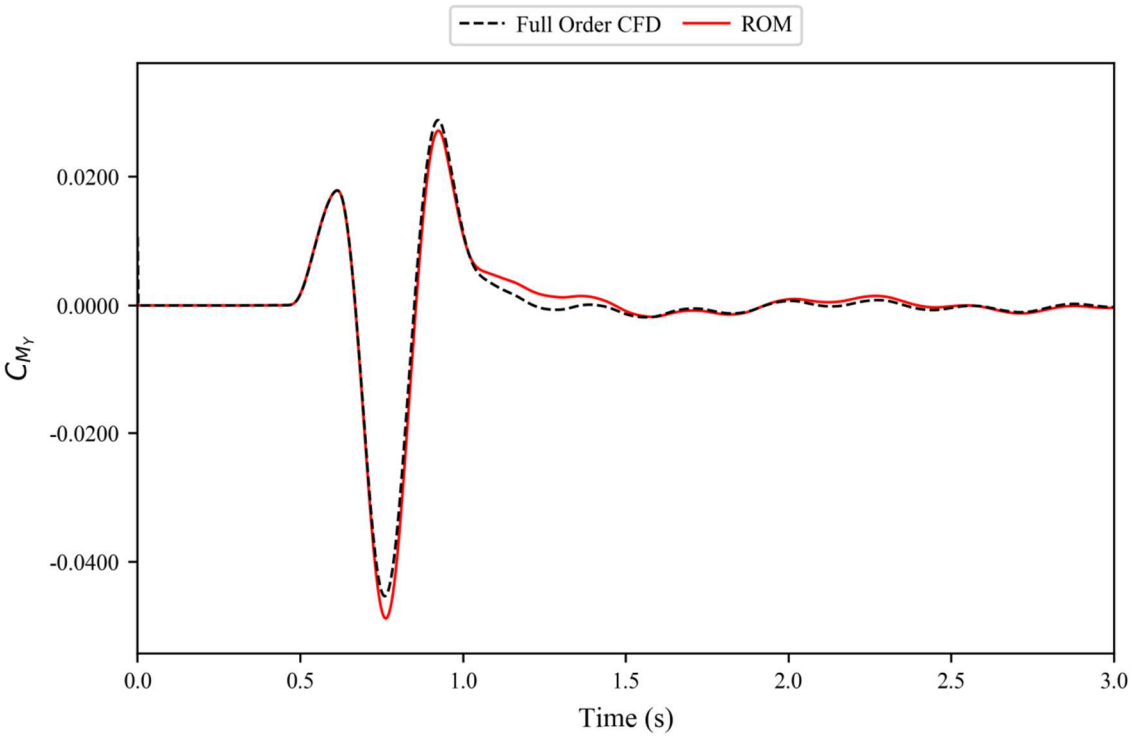

Figure 6.48. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 5, gust case 2.

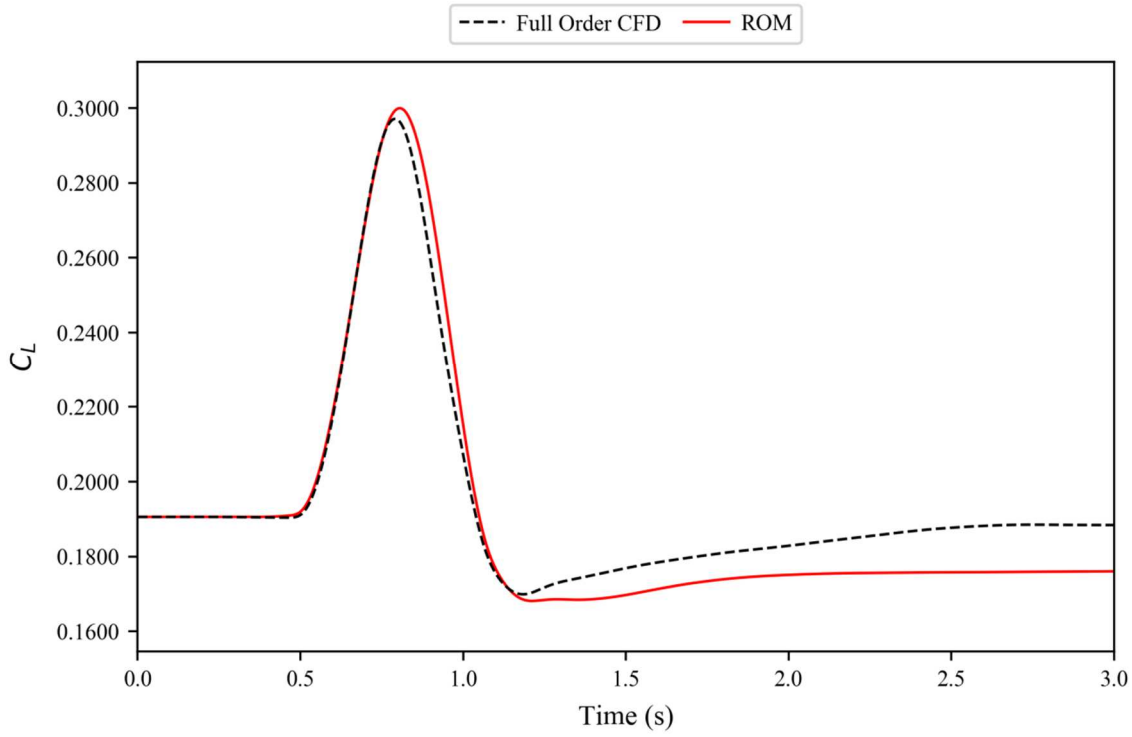

Figure 6.49. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 5, gust case 3.

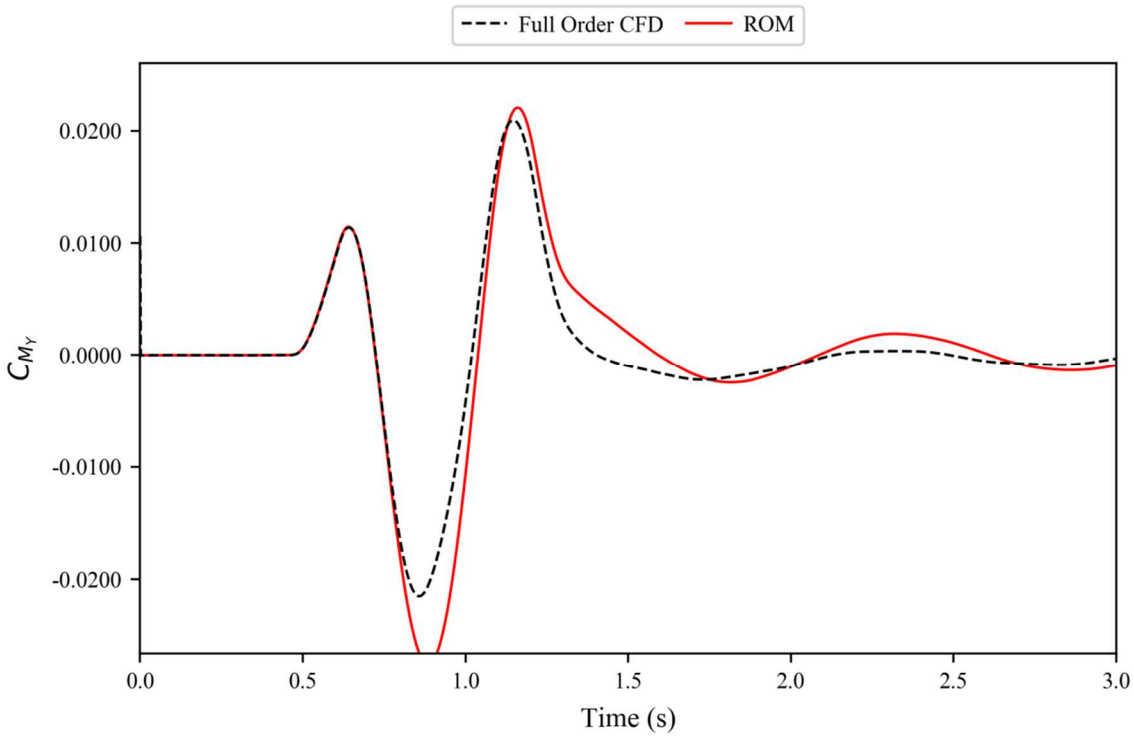

Figure 6.50. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 5, gust case 3.

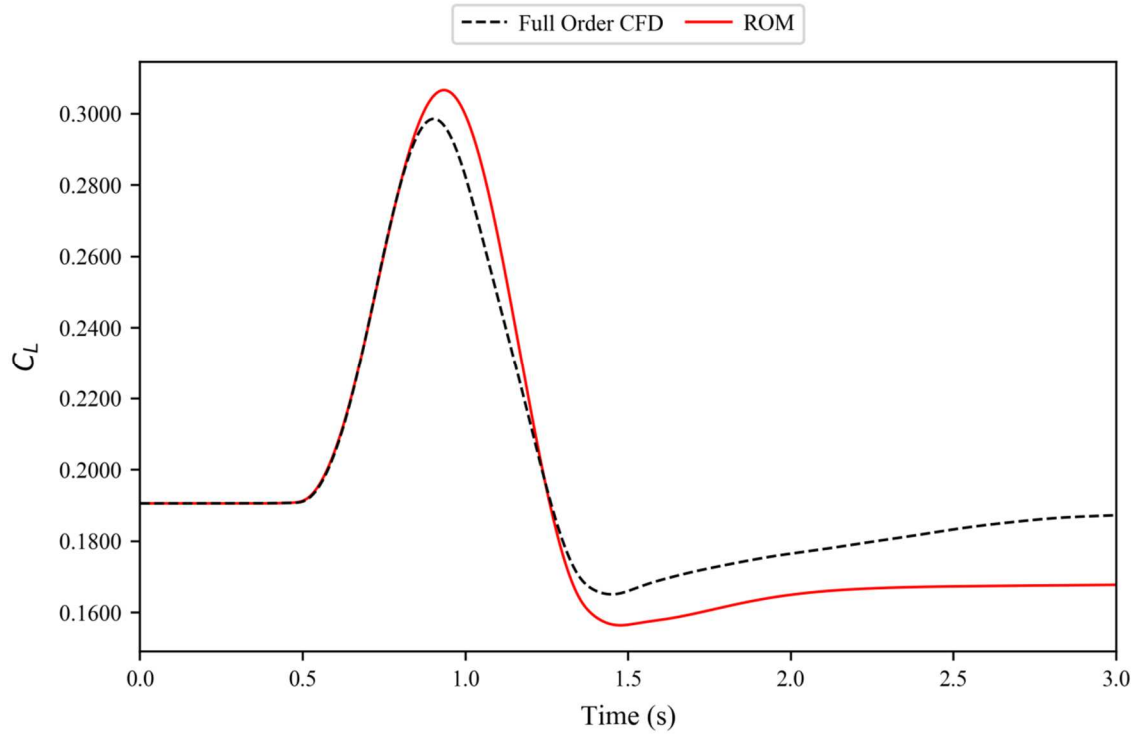

Figure 6.51. ROM results against full order CFD simulation for the coefficient of lift for the flight mechanics model at flight point 5, gust case 4.

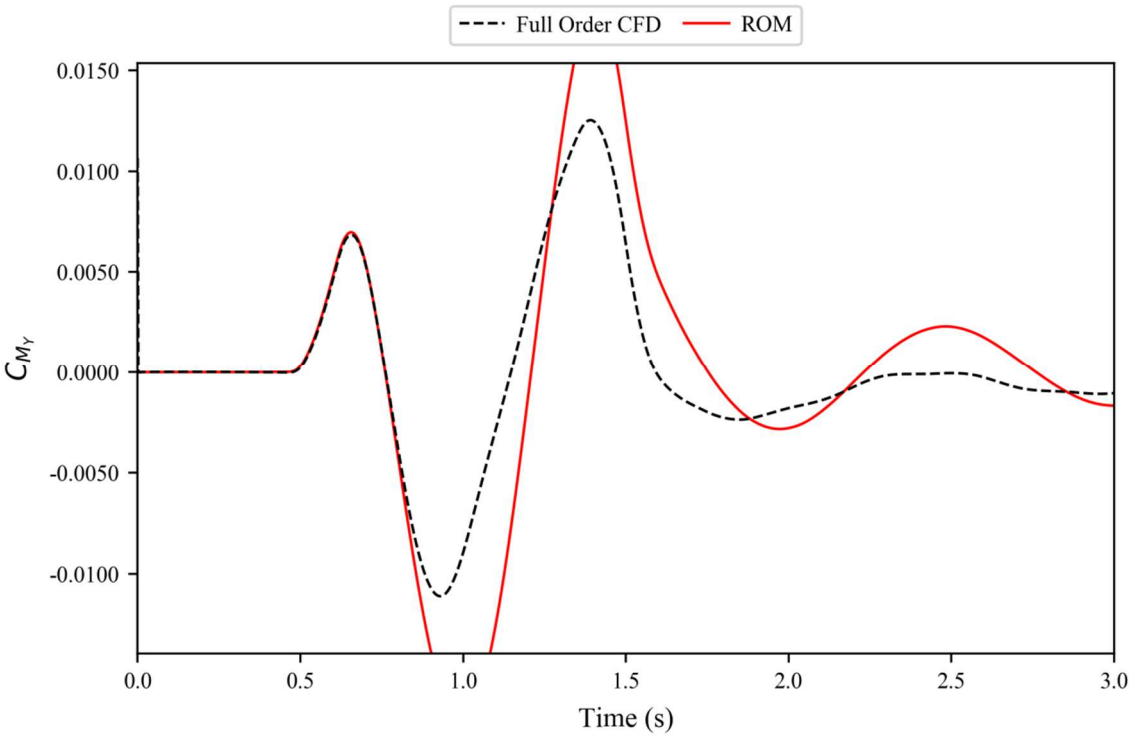

Figure 6.52. ROM results against full order CFD simulation for the coefficient of pitching moment for the flight mechanics model at flight point 5, gust case 4.

6.3.6. Conclusions

The cause of the decrease in accuracy as the gust length increases is likely due to an increase in the non-linearities; which would explain why the sailplane case experiences a higher drop in accuracy compared to the wing and generic wide-bodied aircraft models (with the rigid-body degrees of freedom likely increasing the presence of non-linear behaviour). Whilst not a definitive way of checking for non-linearities, it is worth looking at the coefficients of pressures on the top surfaces of the aircraft for gust case 4 at flight point 3; for before gust impact (Figure 6.53), at the peak of the gust (Figure 6.54) and after the gust has passed (Figure 6.55). From these it can be seen that there are no strong shock waves forming on the lifting surfaces and so the non-linearity is likely to be being caused by a different mechanism.

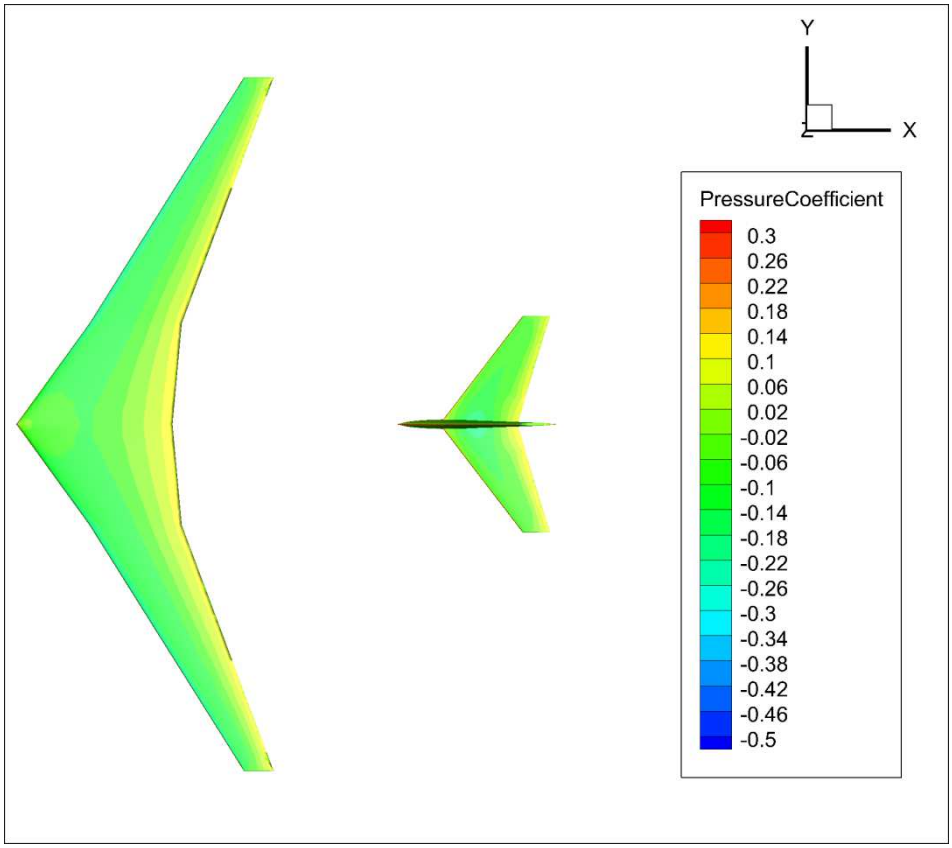

Figure 6.53. Coefficients of pressure on the top of the simple aircraft model from before gust impact for gust 4 for flight point 3.

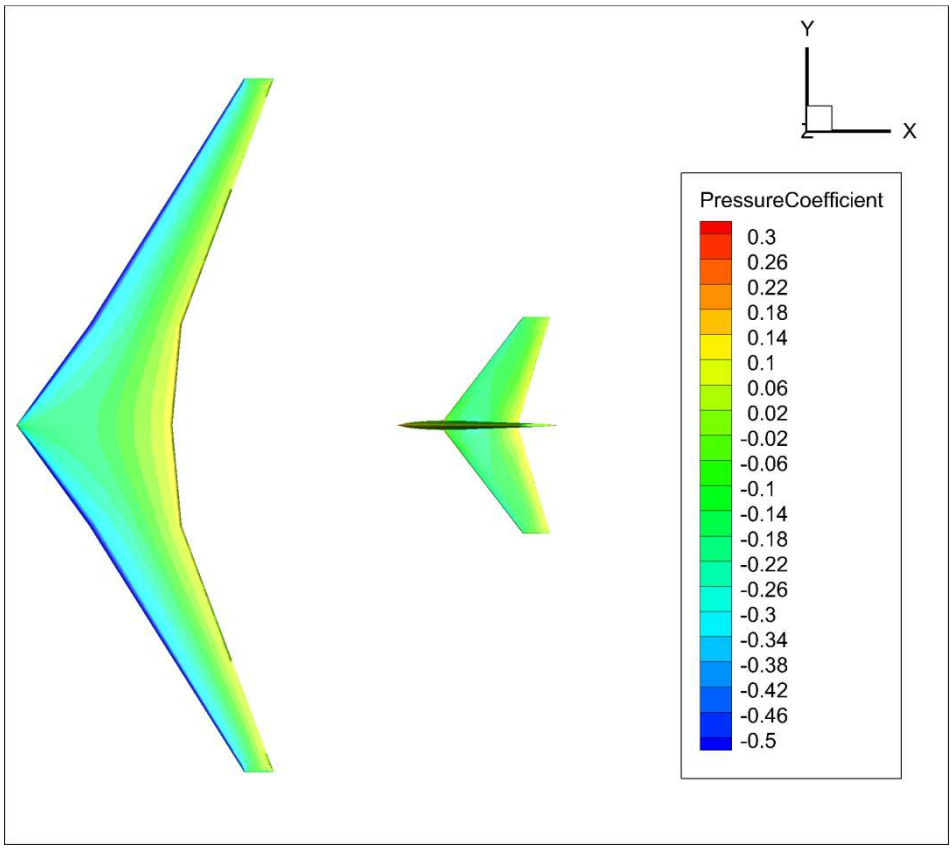

Figure 6.54. Coefficients of pressure on the top of the simple aircraft model at the peak of gust case 4 for flight point 3.

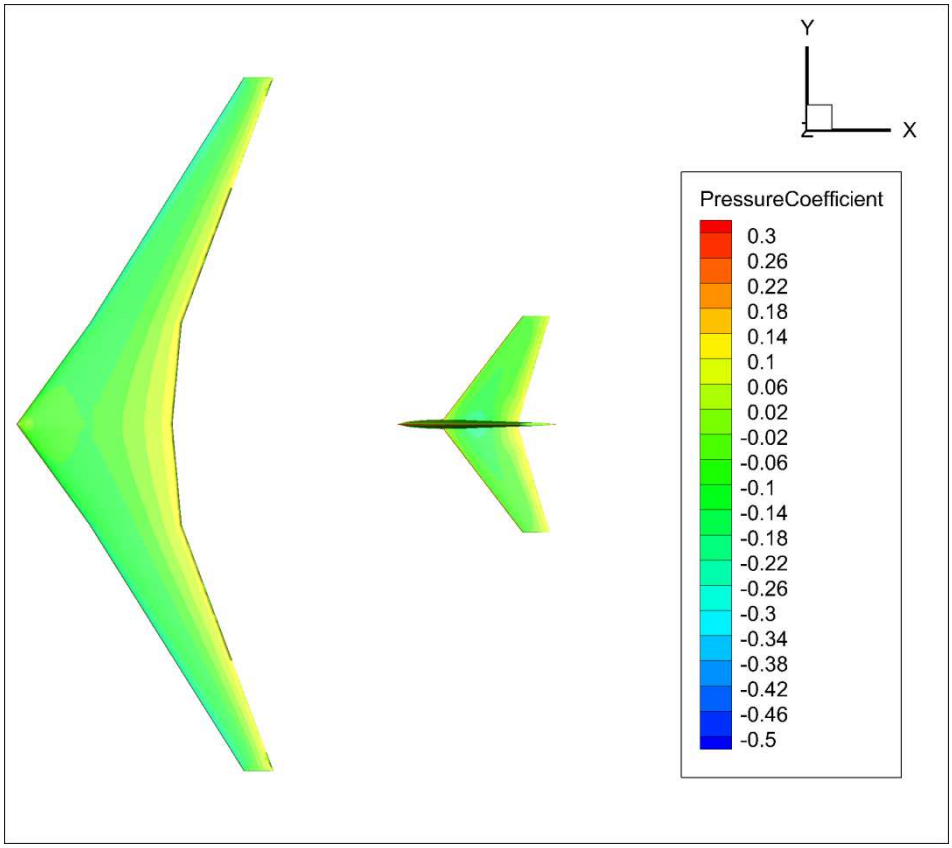

Figure 6.55. Coefficients of pressure on the top of the simple aircraft model after the gust has passed for gust case 4 at flight point 3.

Additionally, the effect of the gust on trim should also be noted. Whereas for all other cases, the model is fixed and thus starts and ends at the same angle of attack, the nature of the 6 degrees of freedom means this isn't the case here. Whilst in some instances the angle of attack does return to approximately the starting value (see Figure 6.56), it should be noted that for some of the larger gust cases the effect of the gust causes the aircraft to effectively change to a different trim angle; meaning the final angle of attack may differ from the starting value (see Figure 6.57).

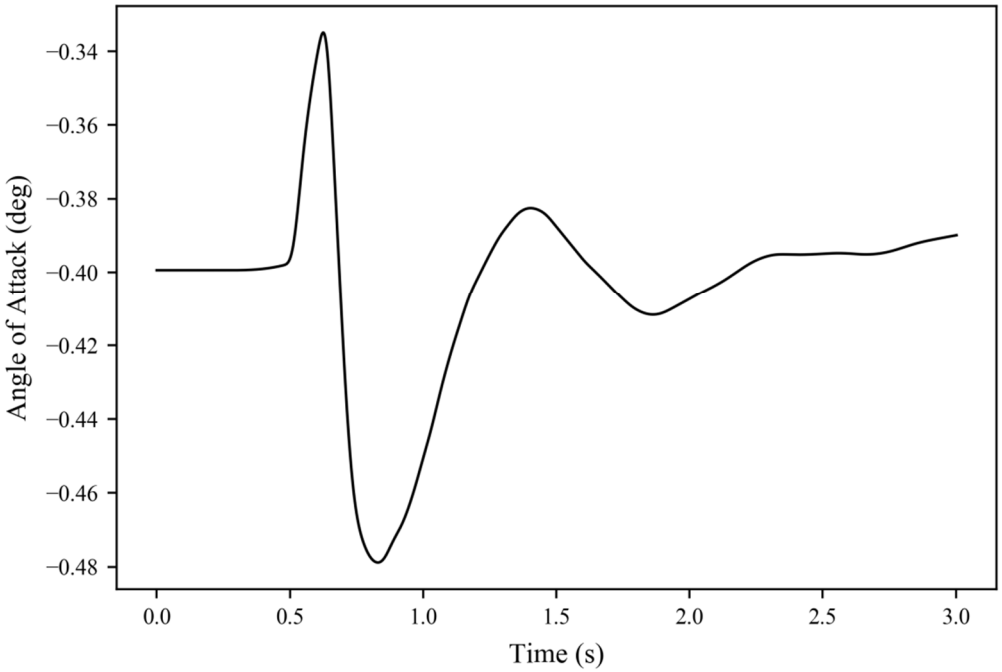

Figure 6.56. Angle of attack history for the flight mechanics model at flight point 2, gust case 1.

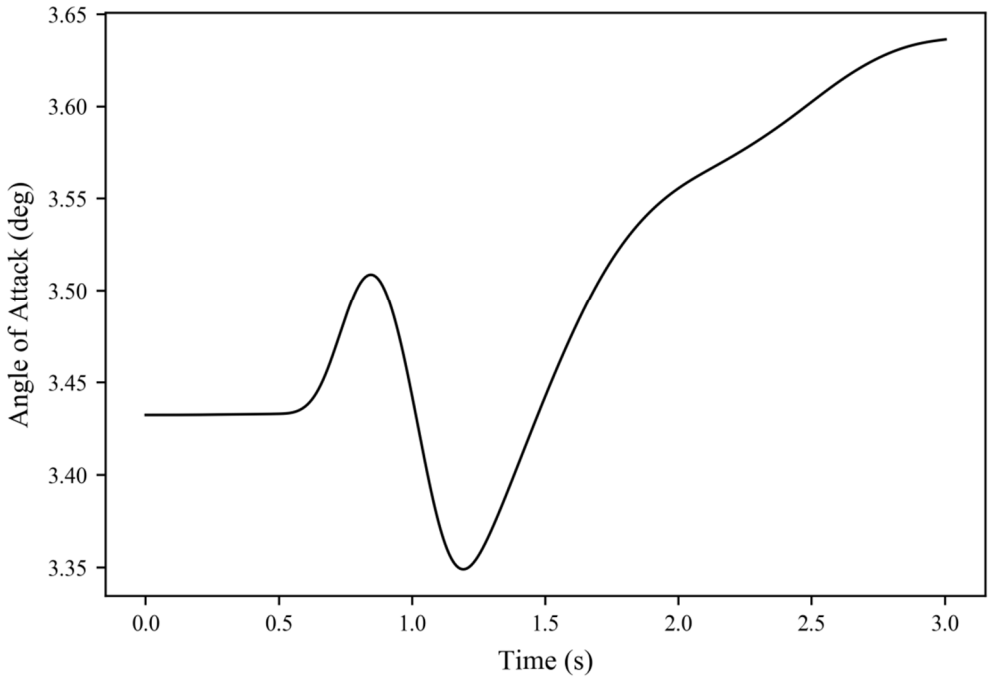

Figure 6.57. Angle of attack history for the flight mechanics model at flight point 5, gust case 3.

Whilst these results do suffer from a reduction in accuracy for the larger gust cases, it should be noted that as no gust alleviation was applied these can be considered worse case scenarios. As such, and as the results for the short to medium length gusts is still typically very good, the ROM can be viewed as still useful for these more complex

cases; albeit with the caveat that additional testing and development would be advisable for some applications. For example; if active flight control was also included the longer gust cases, may produce more linear responses, improving the performance of the ROM (relative to CFD).

It should also be noted that, in conjunction with previous ROM results of the inviscid, rigid wing model and the detailed, rigid, viscous aircraft model (see Chapters 0 and 0 respectively), it can be assumed that the ROMs put forward are likely to work on most aerospace type geometries.

7. ROM Adaptations and Additional Applications

It is possible to extend the usefulness of the ROM methods put forward in this thesis by considering various adaptations. Within this thesis, a ROM adaptation is considered any process that effectively sits on top of the normal ROM process and which extends its usefulness in some meaningful way. Such modifications can be post processes (those carried out after the system matrices have been calculated), or they can occur at some point between the running of the relevant CFD simulations and the calculation of the system matrices. Ultimately, an adaption can be used to carry out additional tasks which allow the ROM to be used in a way other than originally intended; often with little additional computational cost.

The ROM methods put forward in this thesis were designed with the application of use in early design stages of aircraft in mind. However their characteristics, notably their near negligible computational cost once built, mean they may be ideally suited to other applications.

7.1. Altitude Adjustment

If the ROM method from the generic wide-bodied aircraft chapter of the report (see Section 5.4) is considered, then it is fair to say that the computational savings offered by the ROM (versus full order CFD) are very high. However, one limitation is any given ROM is valid only for the flight point it was created at. By modifying the way a ROM is constructed from the CFD data, it is possible to overcome this limitation to some degree by making the ROM valid for any altitude so long as the original Mach number is maintained.

To test this, a ROM (for the aircraft model detailed in Chapter 0) was created at one altitude (sea level) before modifications (see Section 3.5.1) were made to adapt the results at that altitude to allow a ROM to be created for a different altitude. The results were then compared to those of both full order CFD and of a ROM created at the correct altitude.

7.1.1. Results

Here, a ROM was created for flight point 3 for the generic wide-bodied aircraft (see Table 5, Chapter 0) and then a modification to the ROM building process was made to use the same CFD results, but changed to work at flight points 4 and 5. This is then compared to the full order CFD for these two flight points (4 and 5), as well as the ROMs previously put forward for these flight points in Sections 5.4.3 and 5.4.4.

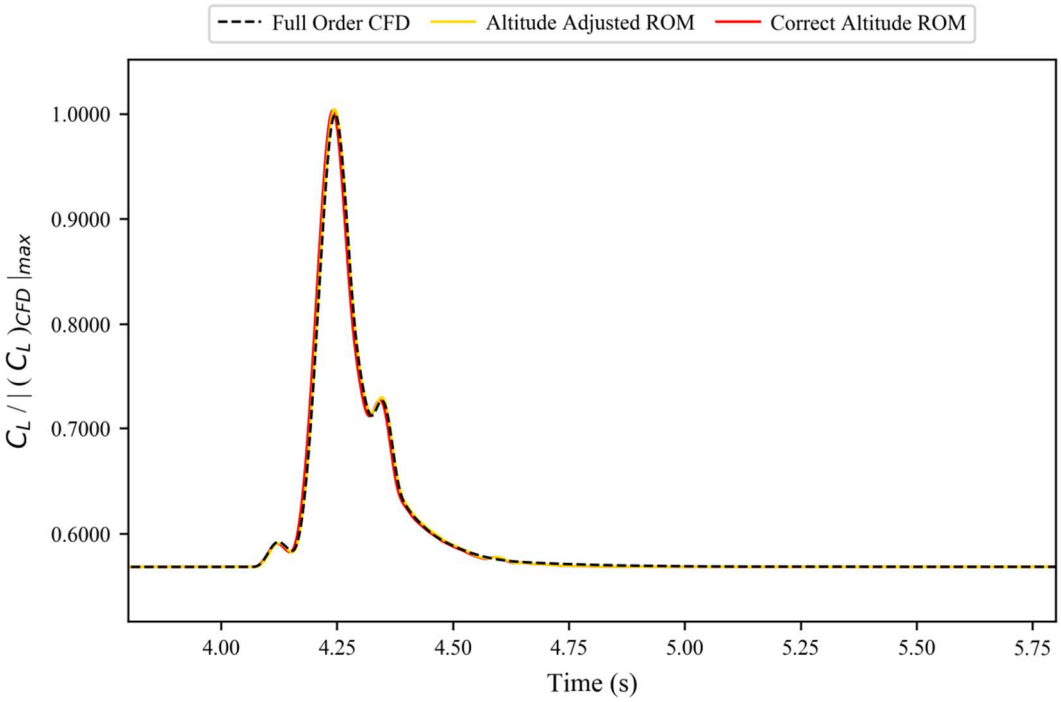

Figure 7.1. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 4, gust case 1.

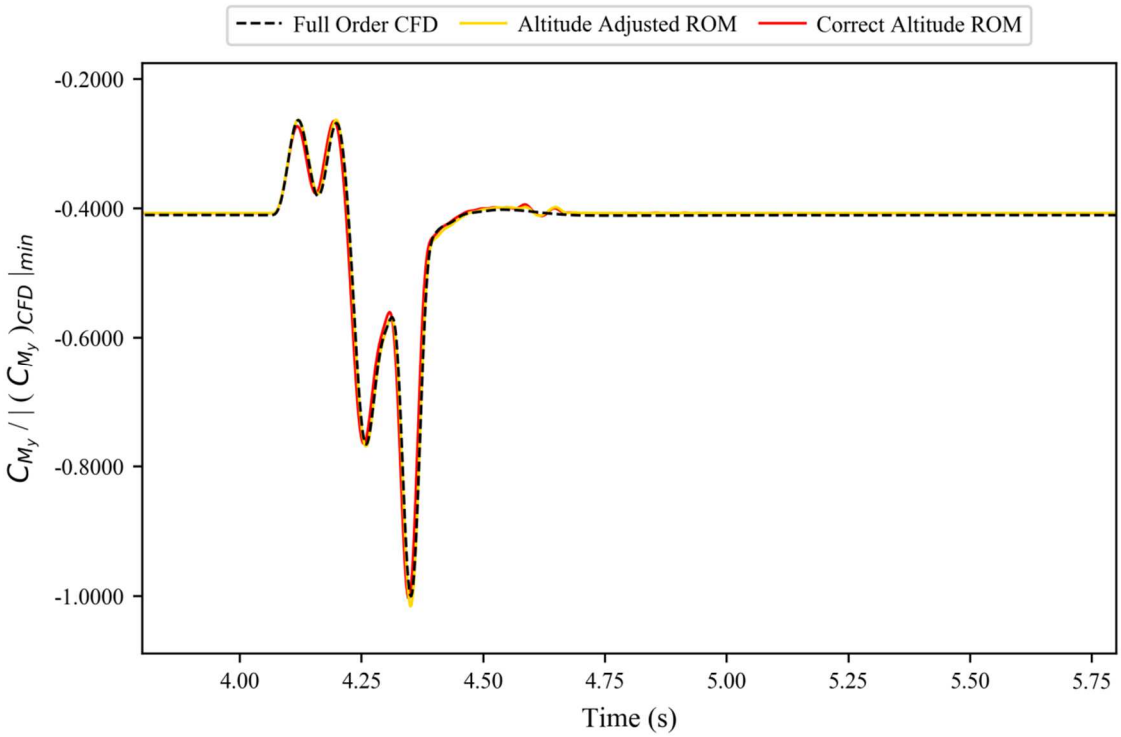

Figure 7.2. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 4, gust case 1.

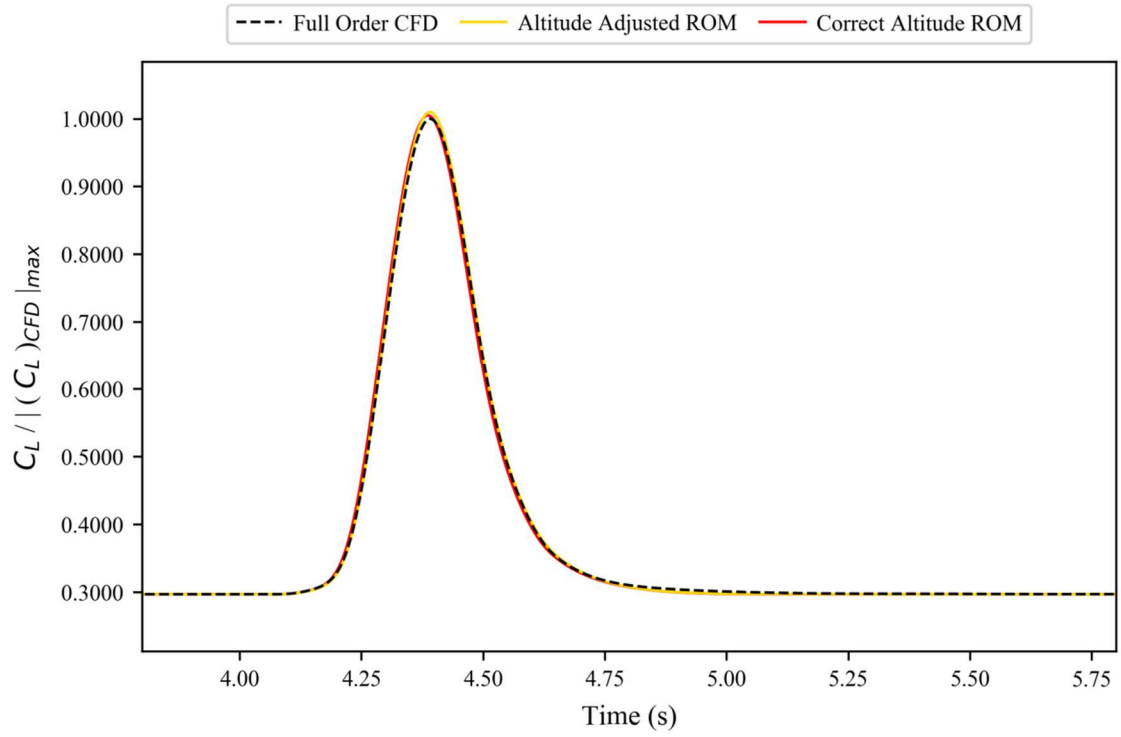

Figure 7.3. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 4, gust case 2.

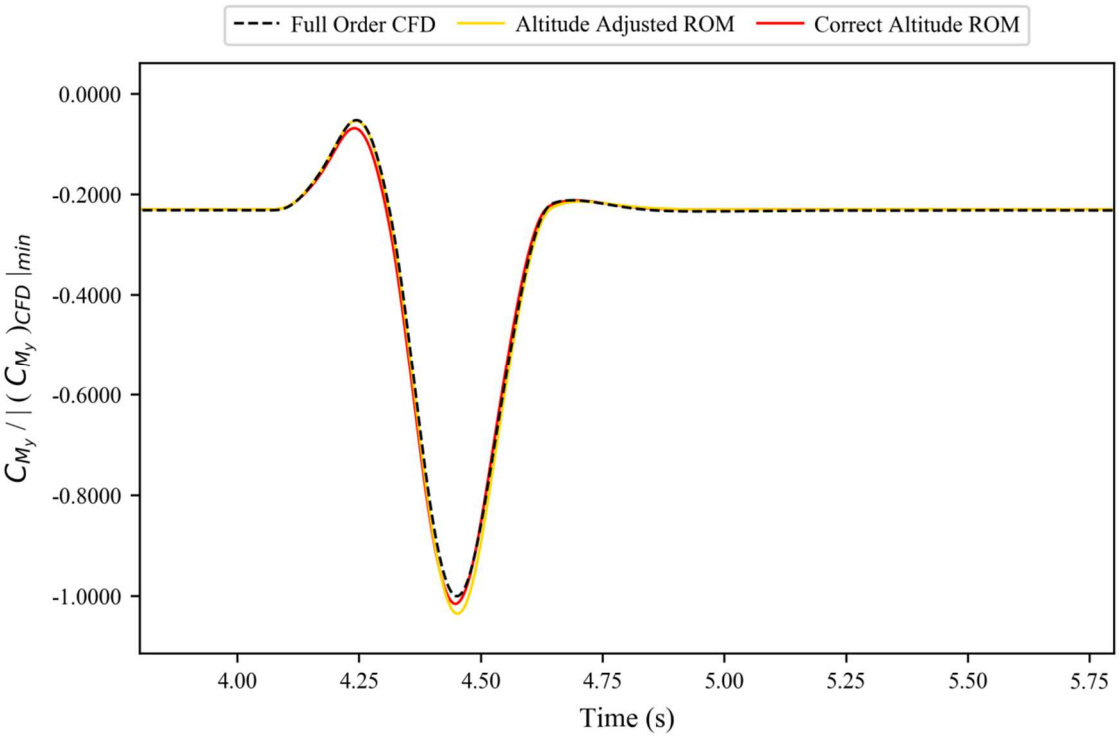

Figure 7.4. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 4, gust case 2.

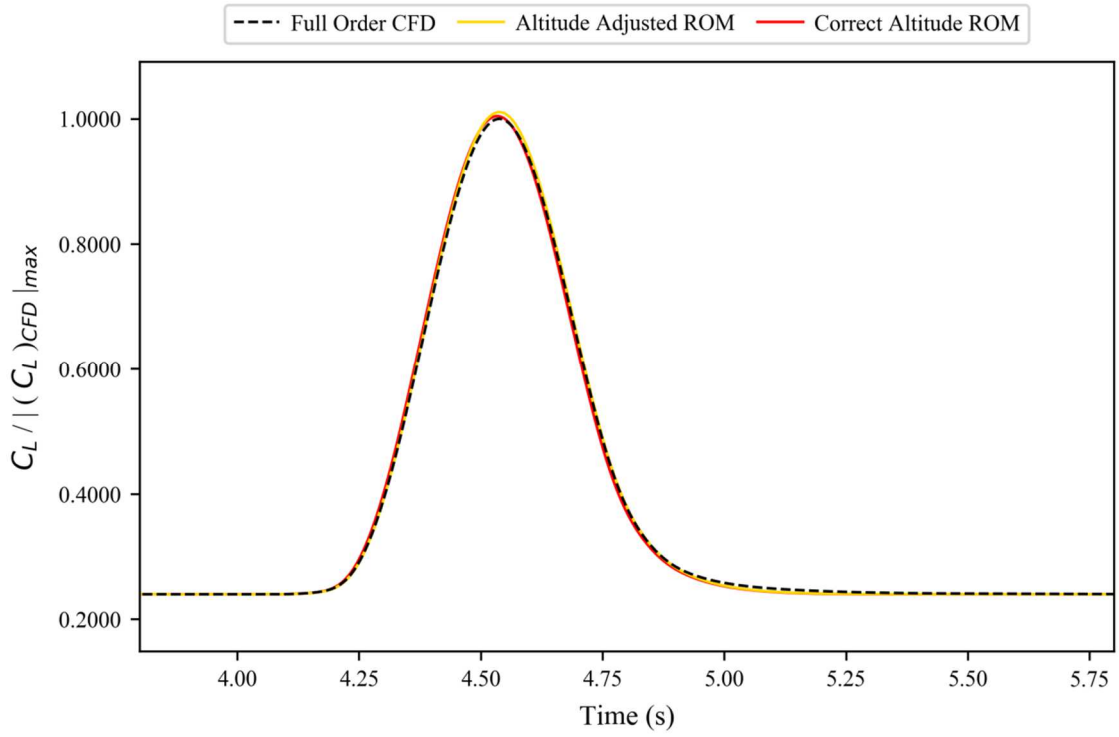

Figure 7.5. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 4, gust case 3.

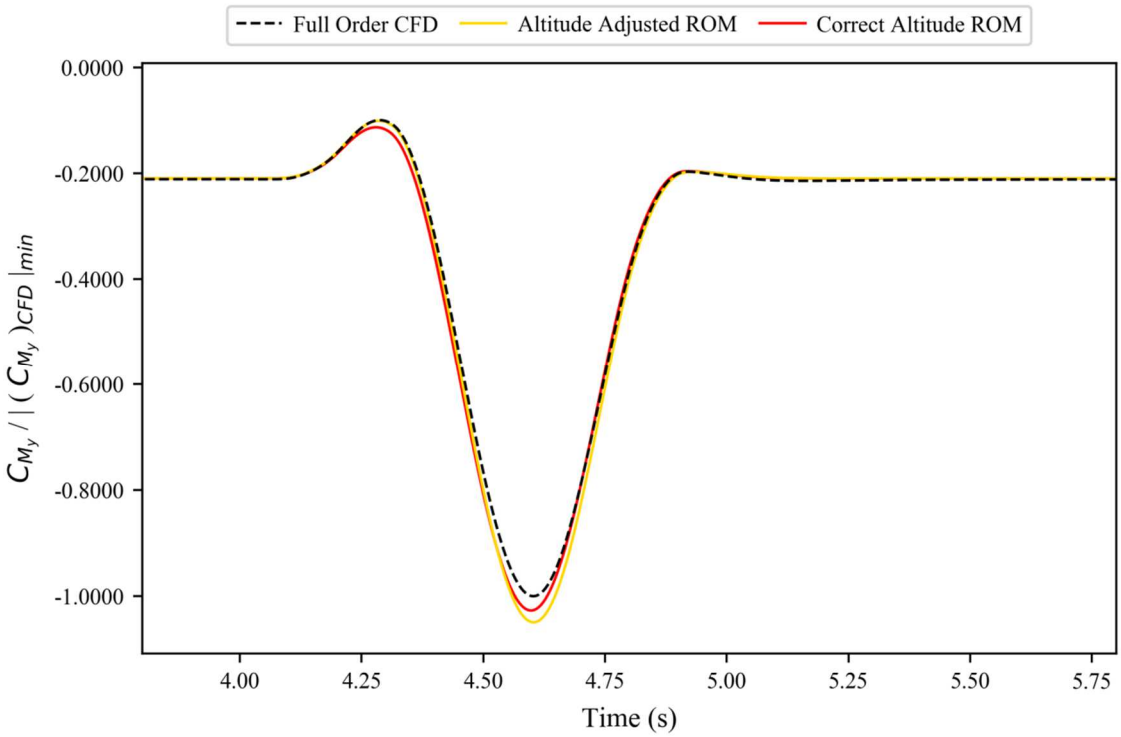

Figure 7.6. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 4, gust case 3.

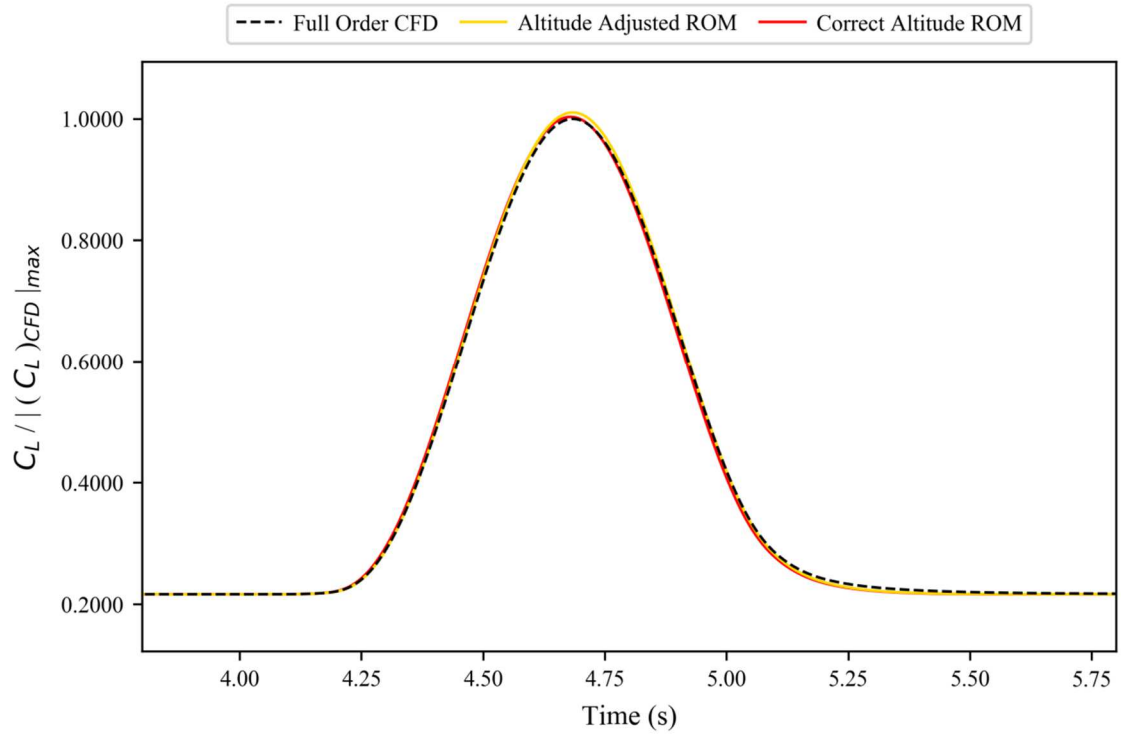

Figure 7.7. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 4, gust case 4.

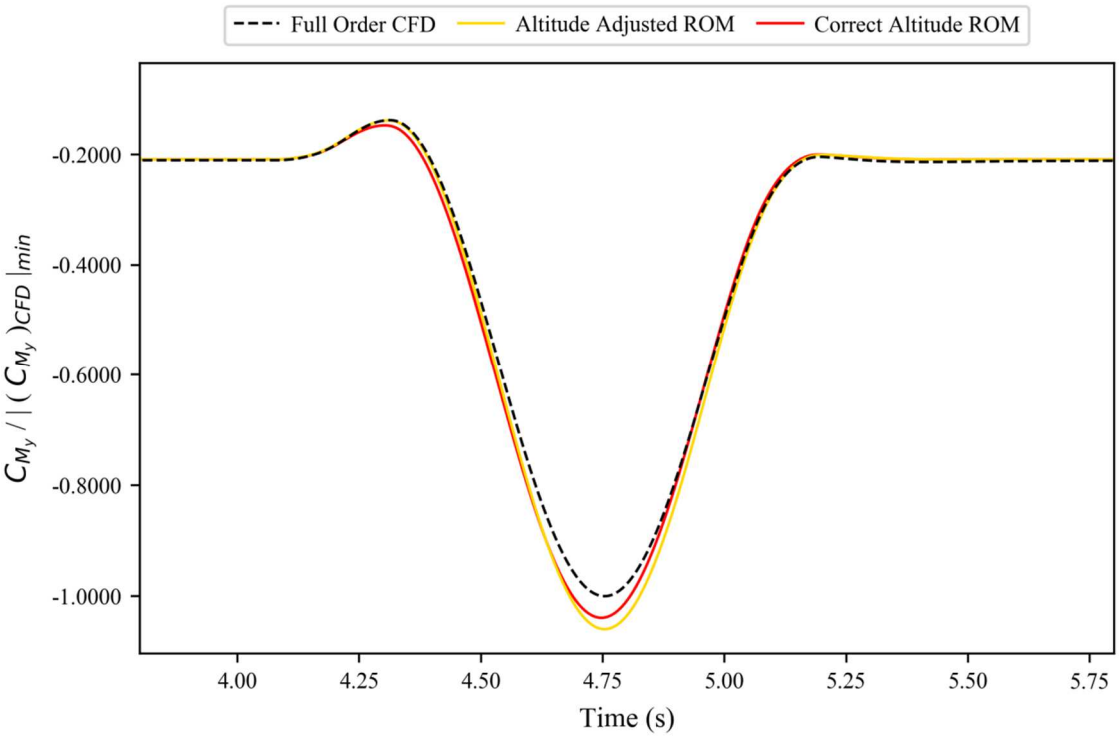

Figure 7.8. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 4, gust case 4.

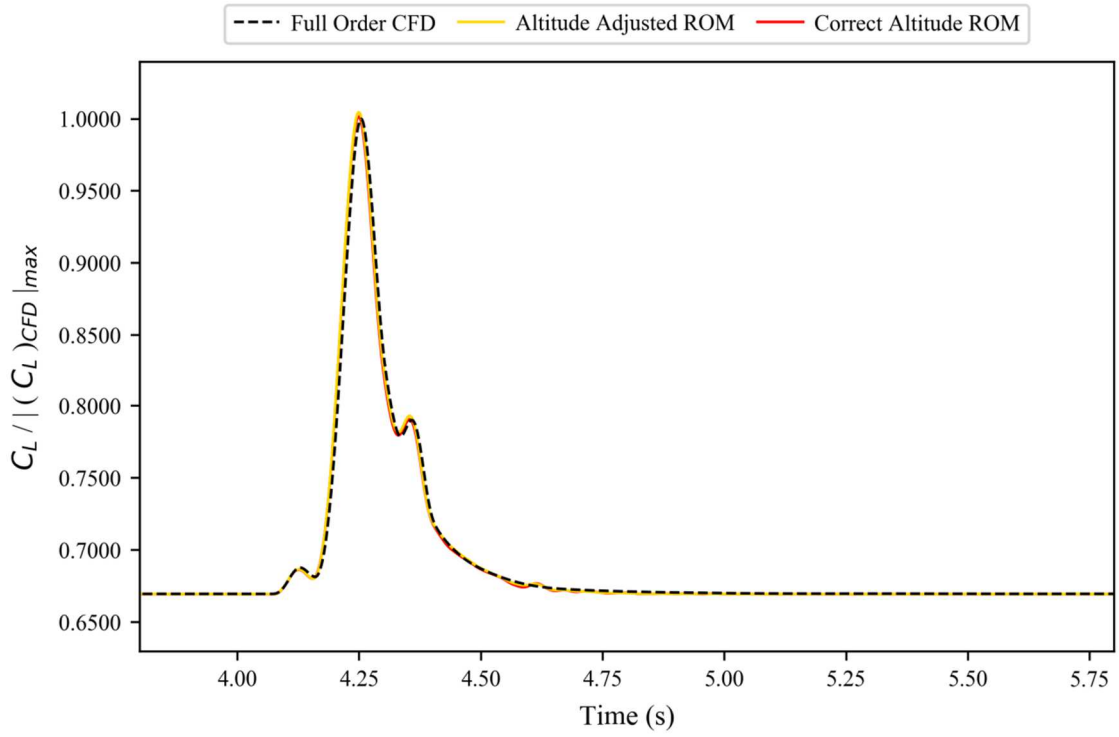

Figure 7.9. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 5, gust case 1.

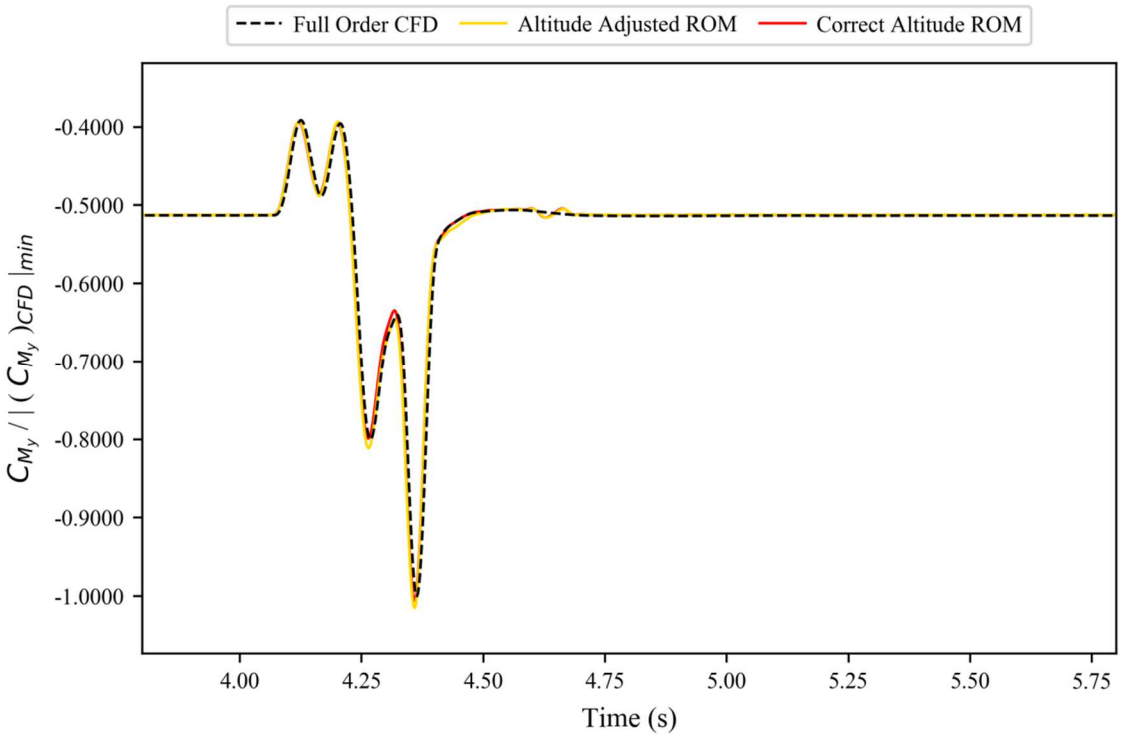

Figure 7.10. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 5, gust case 1.

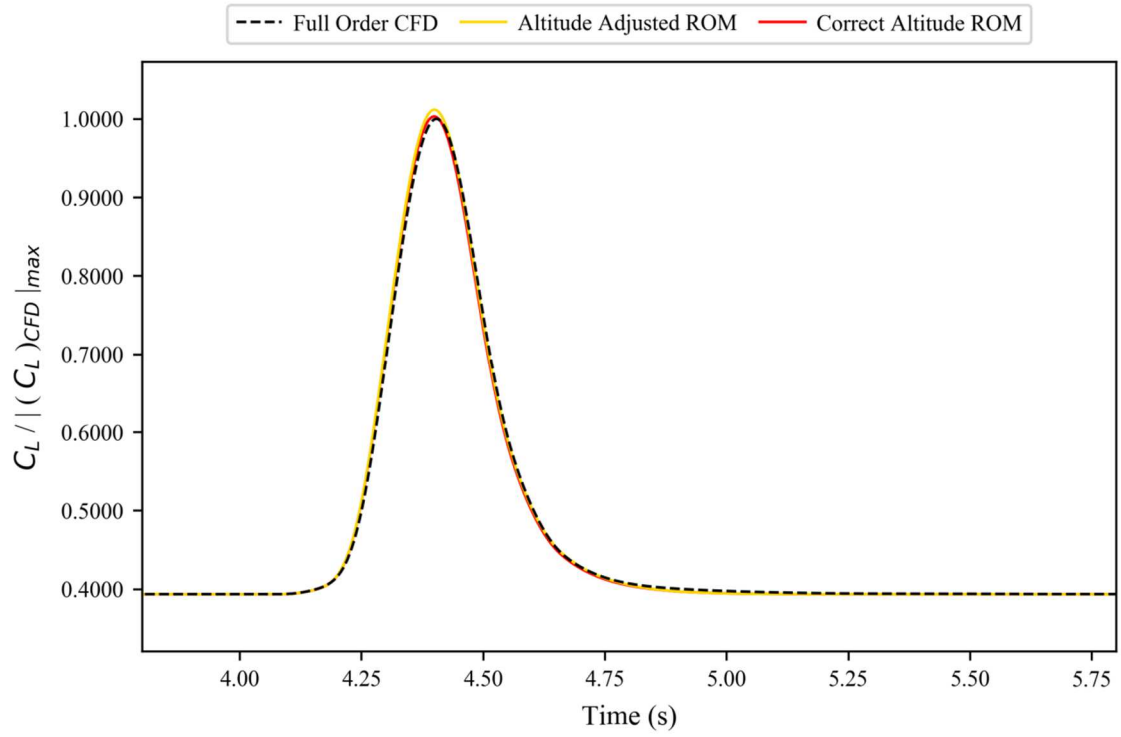

Figure 7.11. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 5, gust case 2.

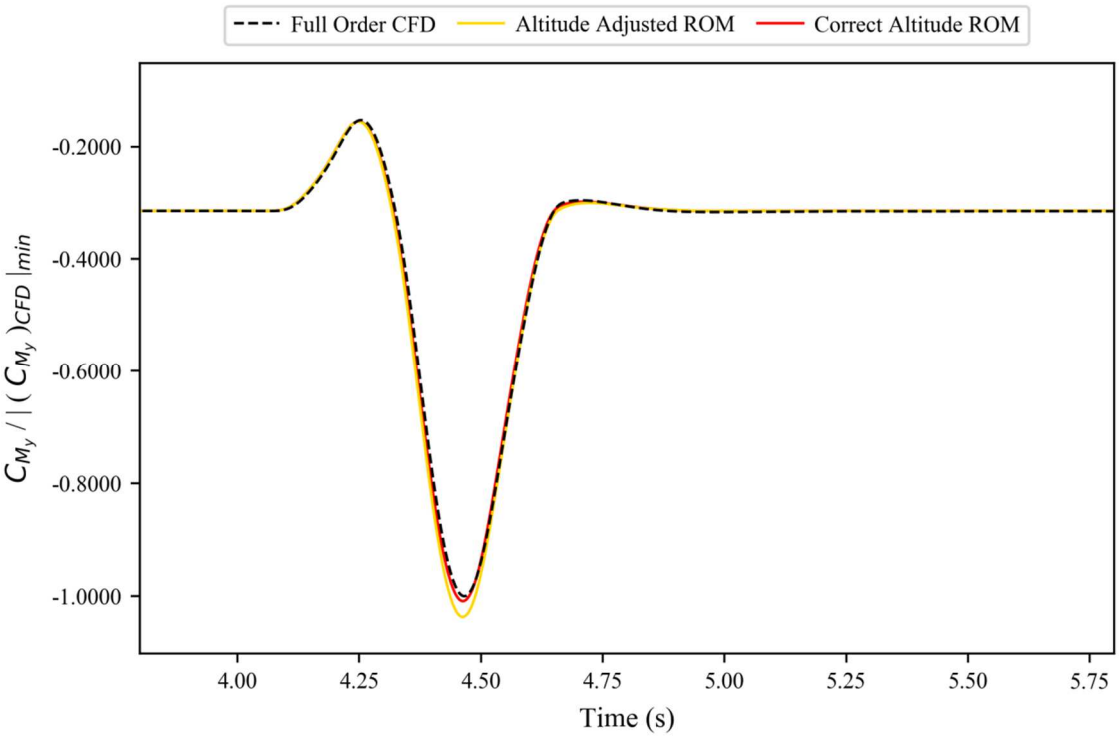

Figure 7.12. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 5, gust case 2.

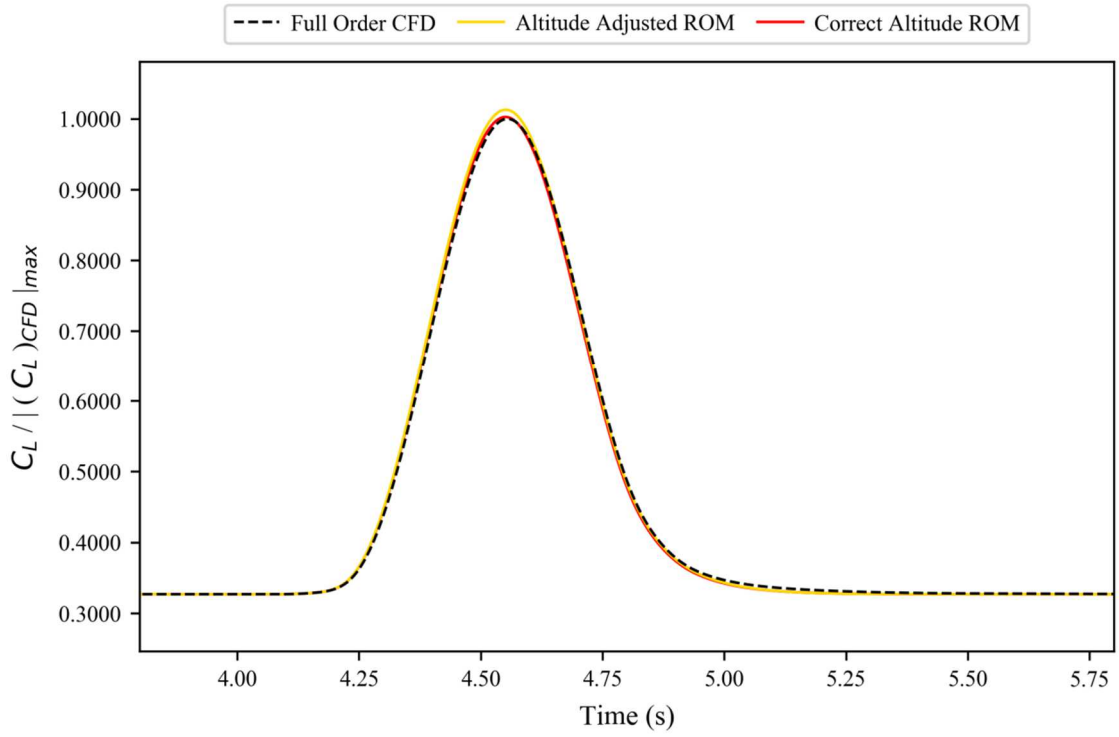

Figure 7.13. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 5, gust case 3.

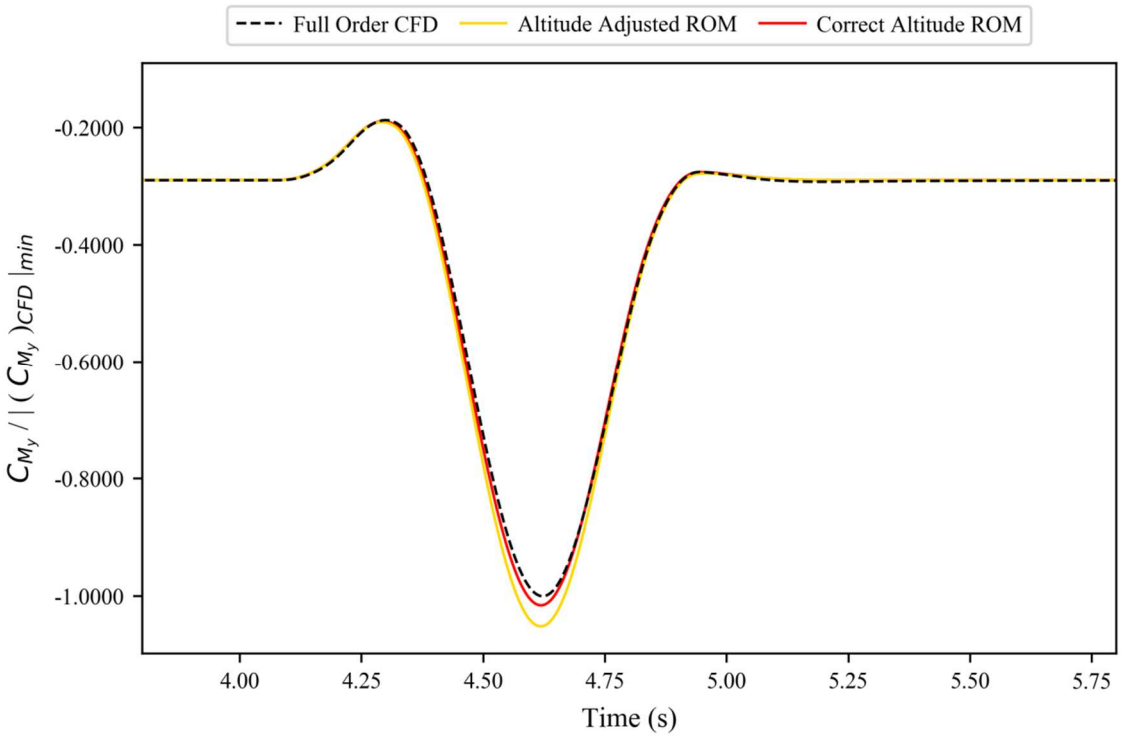

Figure 7.14. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 5, gust case 3.

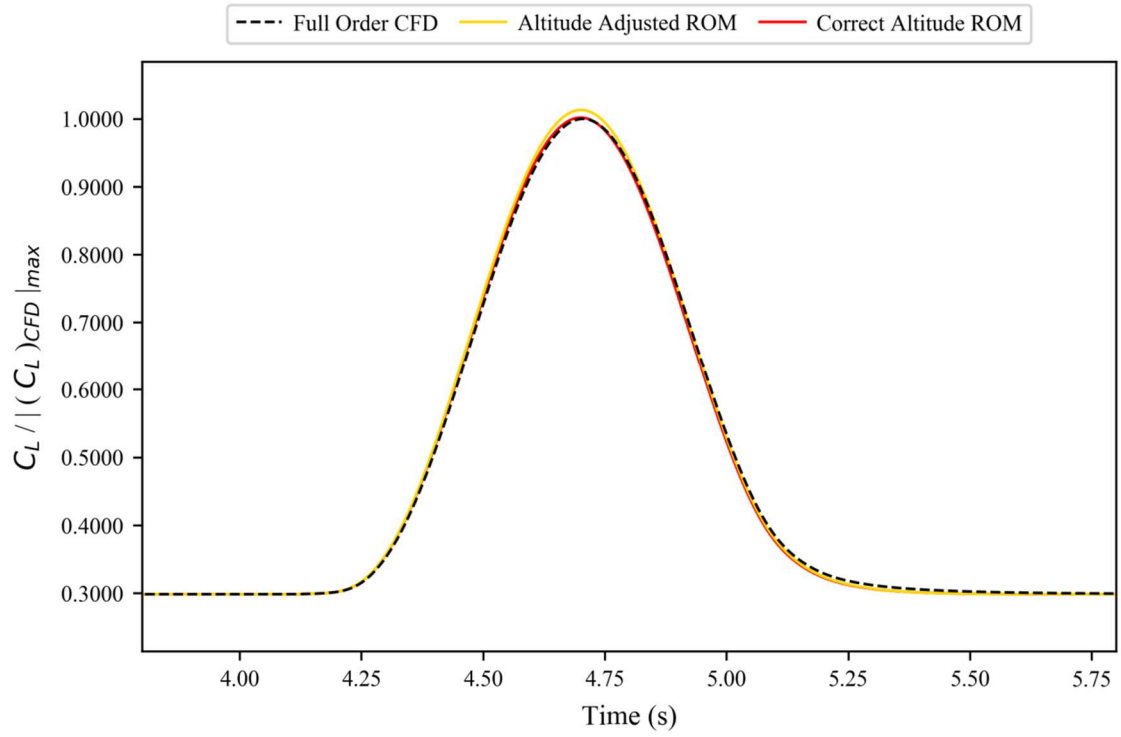

Figure 7.15. Altitude adjusted (normalised) coefficient of lift for the generic wide-bodied aircraft model at flight point 5, gust case 4.

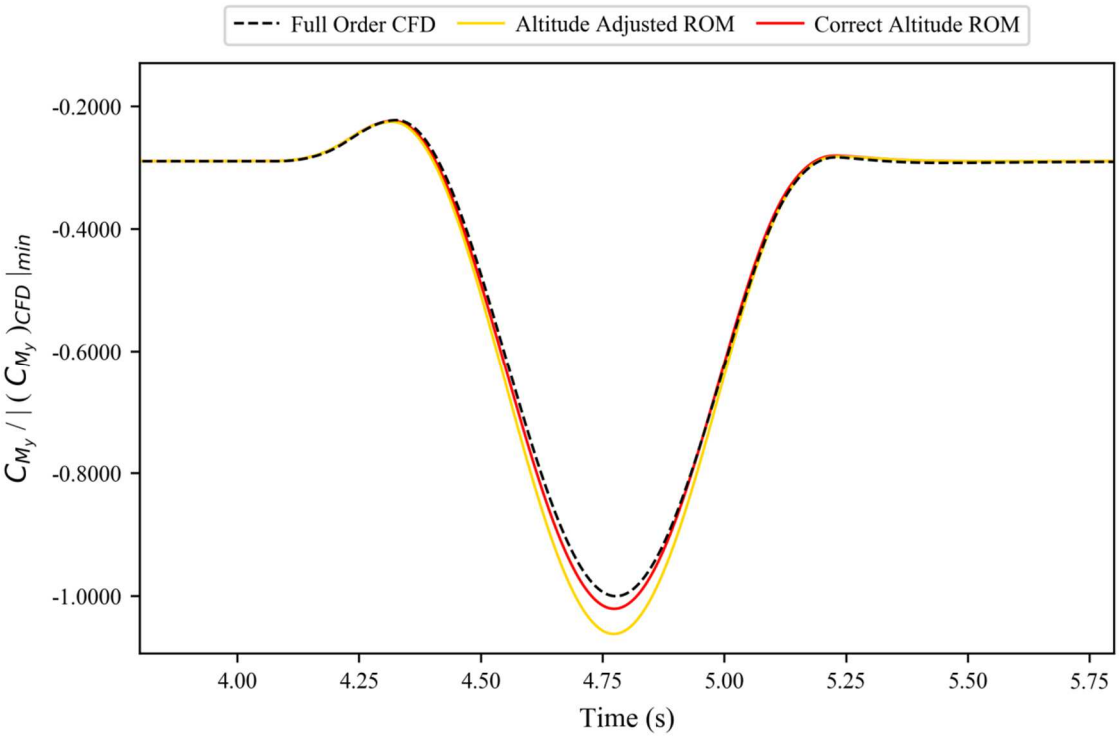

Figure 7.16. Altitude adjusted (normalised) coefficient of pitching moment for the generic wide-bodied aircraft model at flight point 5, gust case 4.

Figures 7.1-7.16 show that the altitude adjustment post process works well. There is a slight degradation in accuracy compared to the original ROM, which is most notable on the longer gust cases and pitching coefficient, however the results are still within a tolerable error margin. Ultimately, this means that in instances where multiple gusts at various altitudes for a given Mach number are desired, and where a slight decrease in accuracy is acceptable, then very large computational savings can be made.

Using the three example flight points (3, 4 and 5) for the generic wide-bodied aircraft, then an idea of the computational savings can be gained. The normal ROM creation method required sharp-edge gusts, etc. to be run at three separate altitudes. However, the altitude adjustment post process allowed a ROM to be created for flight point 3 and then adjusted for the other two flight points, reducing the computational cost by 67% compared to the original method; a considerable amount given that the normal ROM method already offers a very large reduction in computational cost compared to full order CFD.

7.2. Surface Pressure Reconstruction

Whilst the ROM methods put forward within this thesis to this point have focused on obtaining the changes in force coefficients, it is worth noting that in many industrial applications this is not the only output of interest. Often, the surface pressures (either the absolute values or the coefficients) are a desirable output from full order CFD as it can be integrated into other design processes. Therefore, being able to adapt a ROM to achieve this would improve the suitability of it for current industrial applications.

To carry this out, the final (non-modified) ROM method presented in this thesis was taken and the appropriate modifications were made (see Section 3.5.2). The original part of the ROM method would be carried out as normal (producing a time history of a given force coefficient), before the modifications took some of the internal ROM matrices and the surface pressures from the sharp-edged gusts, and used them to output the surface pressures of the gust being modelled. This was then tested on the model (the FFAST wing) and gust cases from Chapter 0.

7.2.1. Sample Surface Pressure Results

One of the best ways to evaluate the resultant surface pressure outputs is to compare them directly to those obtained via full order CFD; as done below. However, as there exists a set of solutions at each time step, for each gust and for the two different force coefficient based ROMs (lift and pitching moment), then it simply isn't possible to present them all within this thesis; instead two cases are presented.

Both cases are at flight point 2, with one being produced using a lift based ROM for the shortest gust (gust case 1) and another being produced using a pitching moment based ROM for the longest gust (gust case 4). For each, a small sampling of time steps was used (see Figures 7.17 and 7.26); from before the gust impacts the model, the initial part of the response, the peak of the response and as the system is settling.

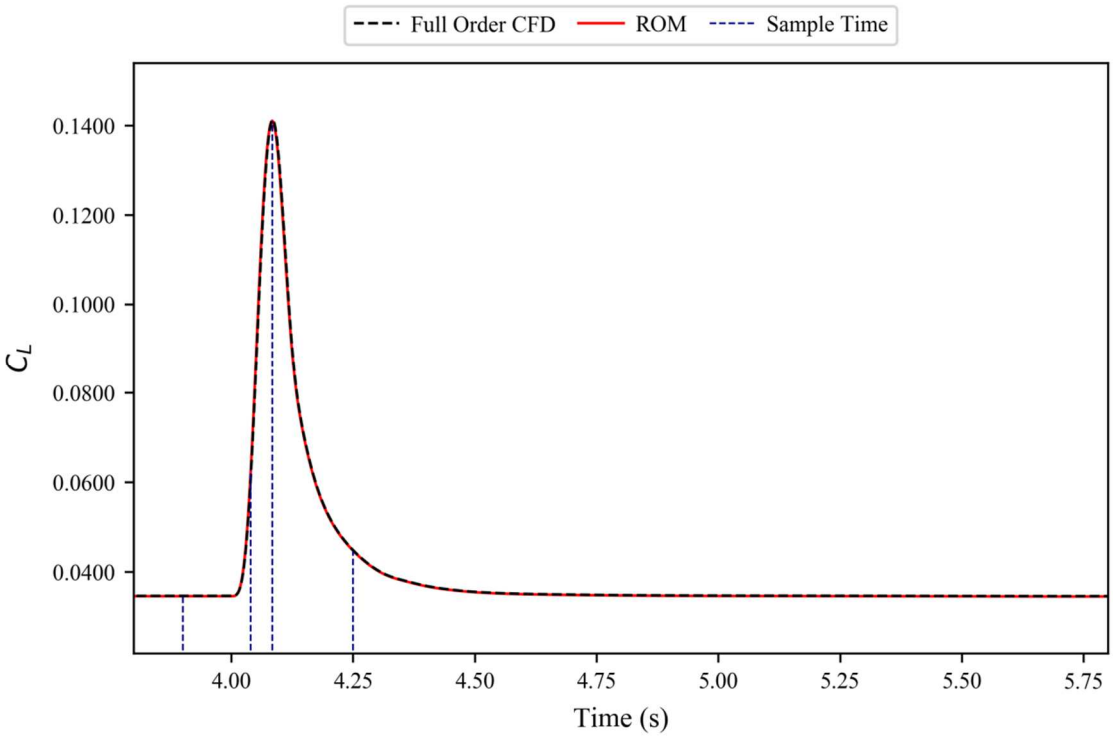

Figure 7.17. Locations of the sample times used for flight point 2, gust case 1.

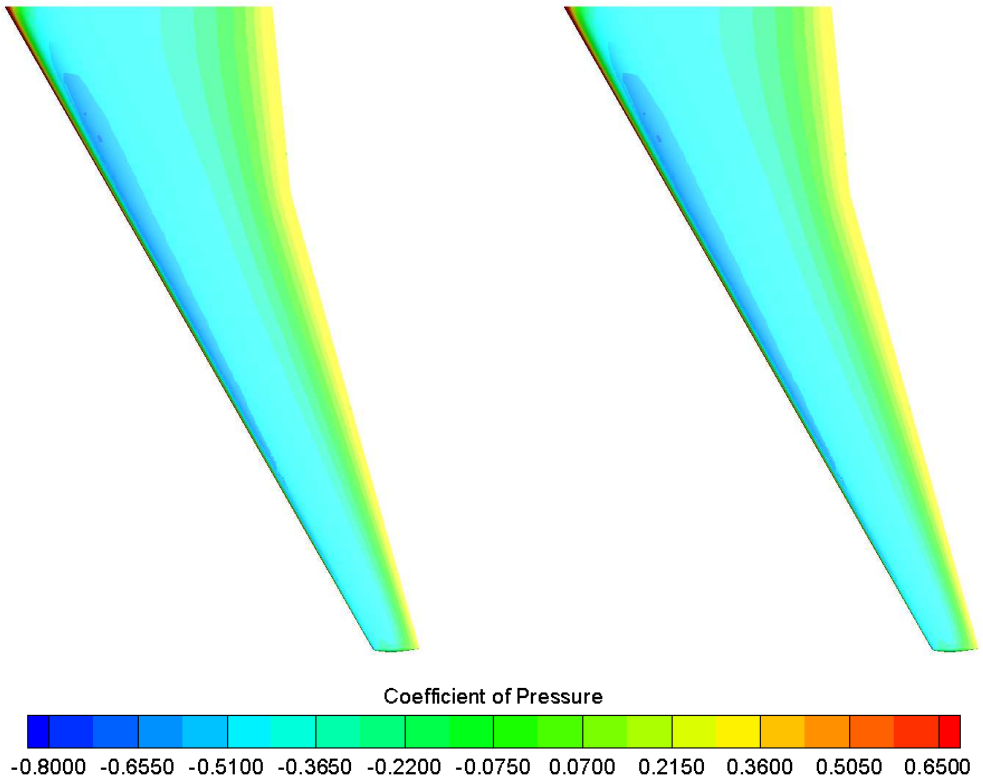

Figure 7.18. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 1 and at 3.900 seconds.

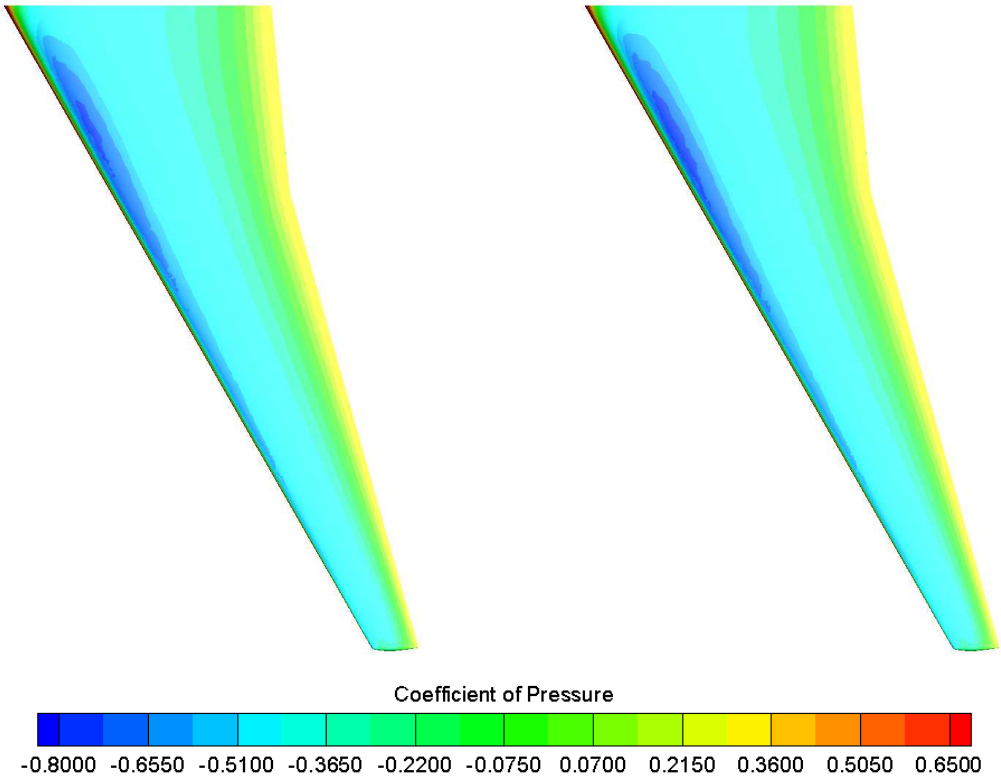

Figure 7.19. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 1 and at 4.040 seconds.

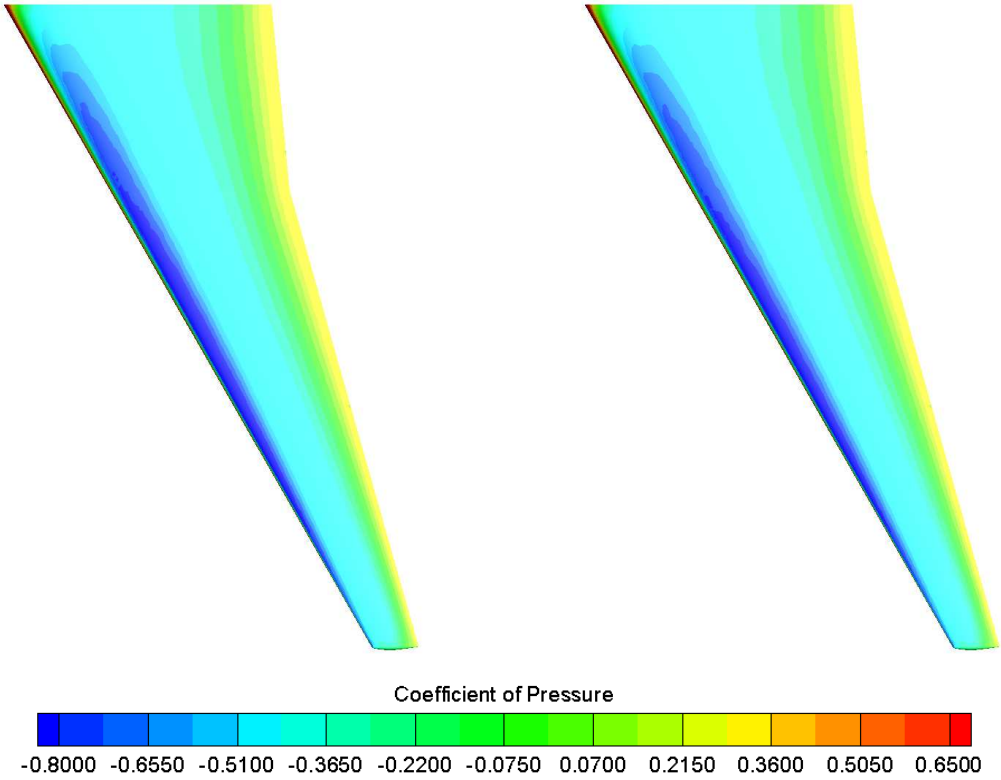

Figure 7.20. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 1 and at 4.084 seconds.

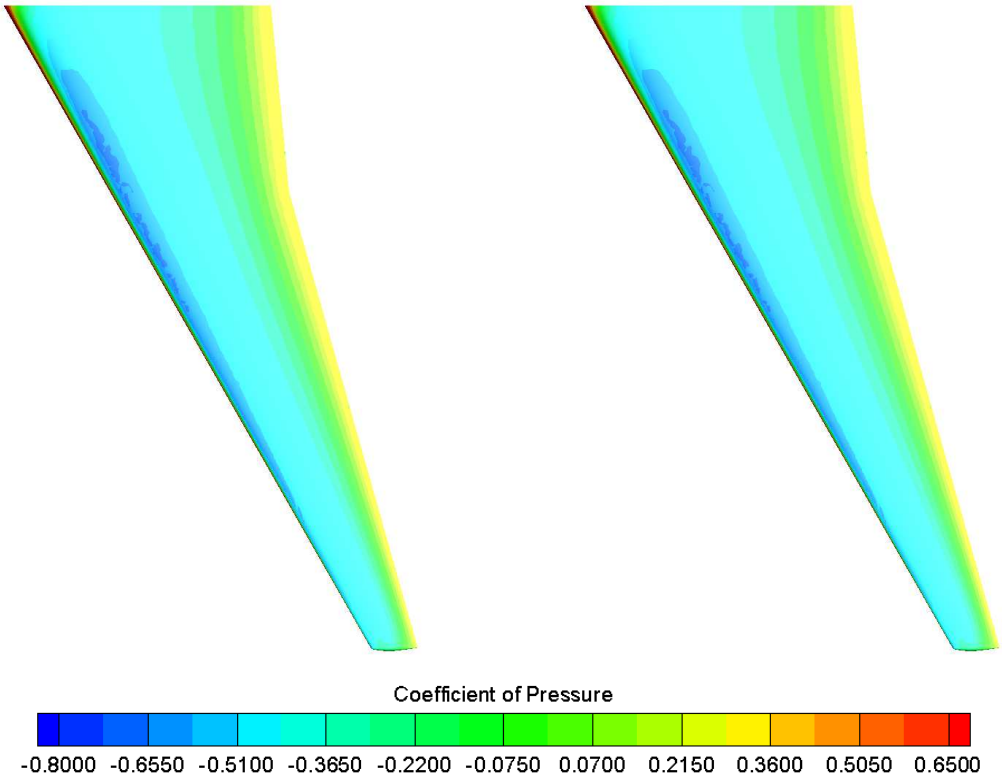

Figure 7.21. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 1 and at 4.250 seconds.

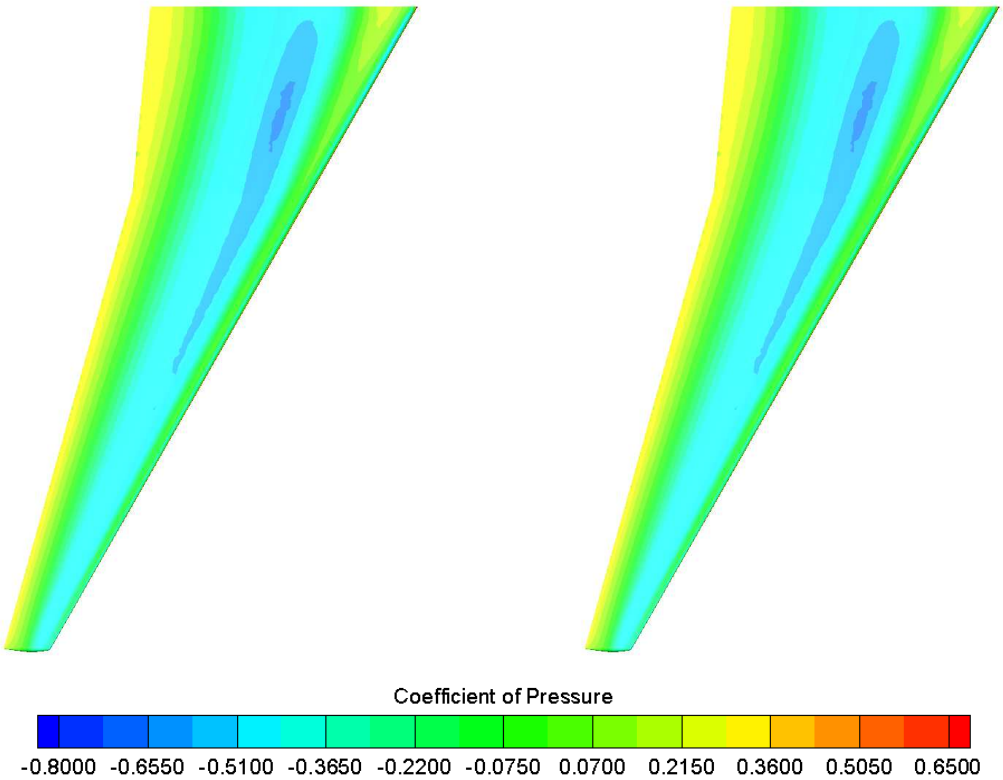

Figure 7.22. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 1 and at 3.900 seconds.

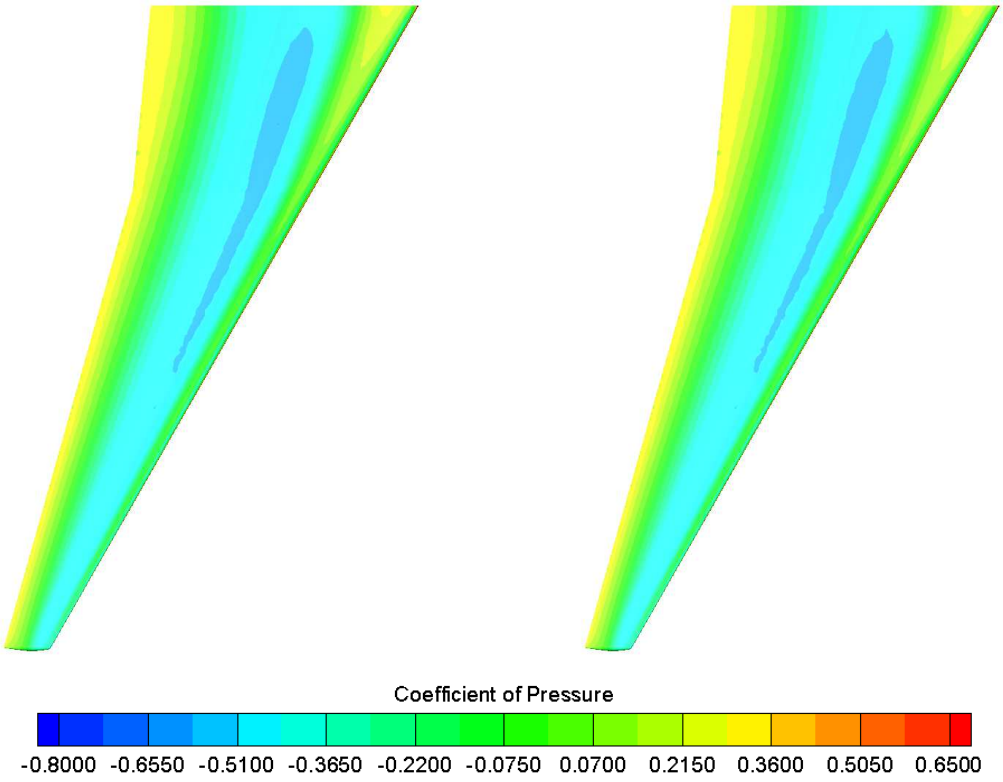

Figure 7.23. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 1 and at 4.040 seconds.

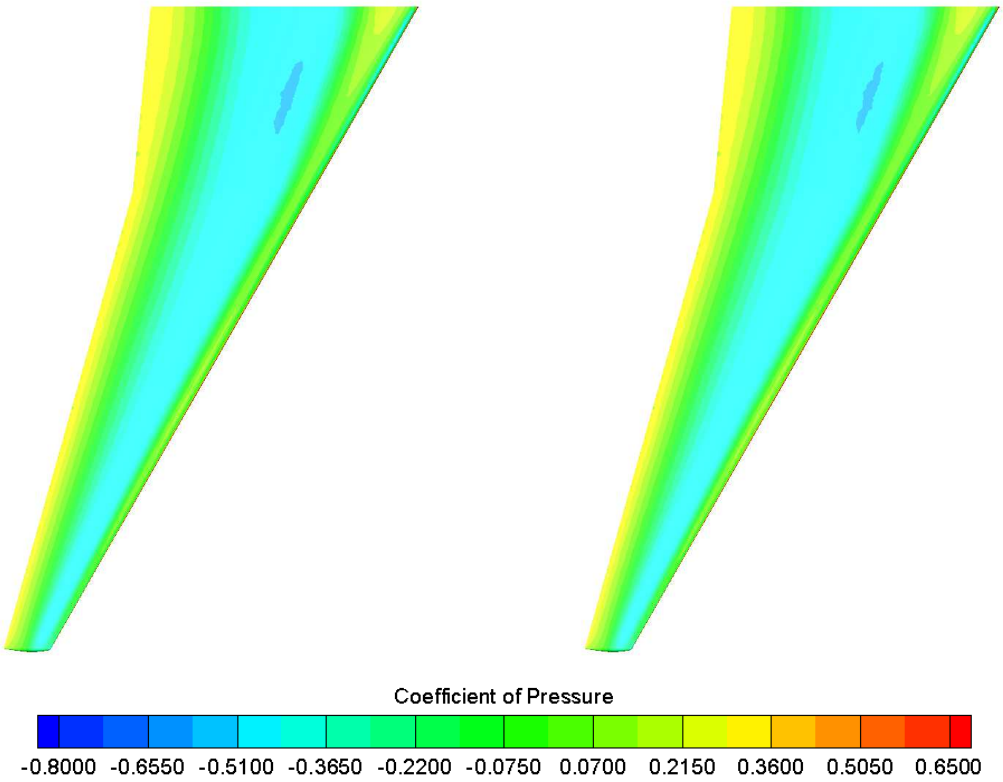

Figure 7.24. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 1 and at 4.084 seconds.

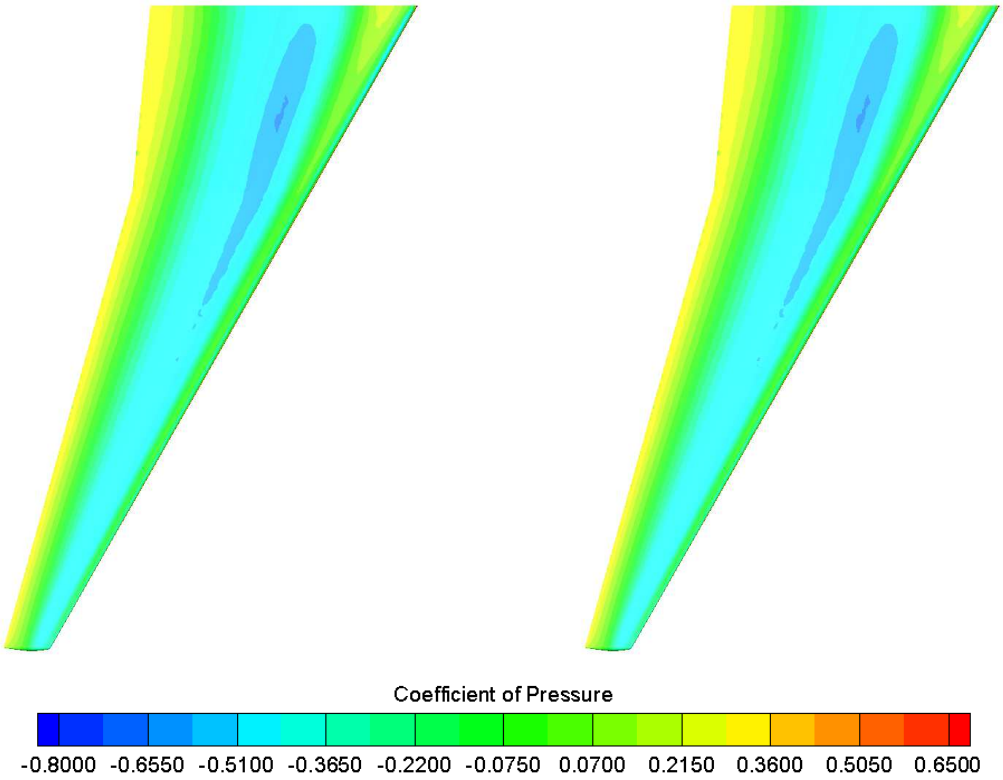

Figure 7.25. Comparison of CFD (left) and a lift based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 1 and at 4.250 seconds.

Figures 7.18-7.25 show a high level of accuracy for the reconstructed surface pressures, compared to those obtained via full order CFD. Whilst there are some slight differences, these are extremely minor and would likely have a negligible impact if used within an industrial process.

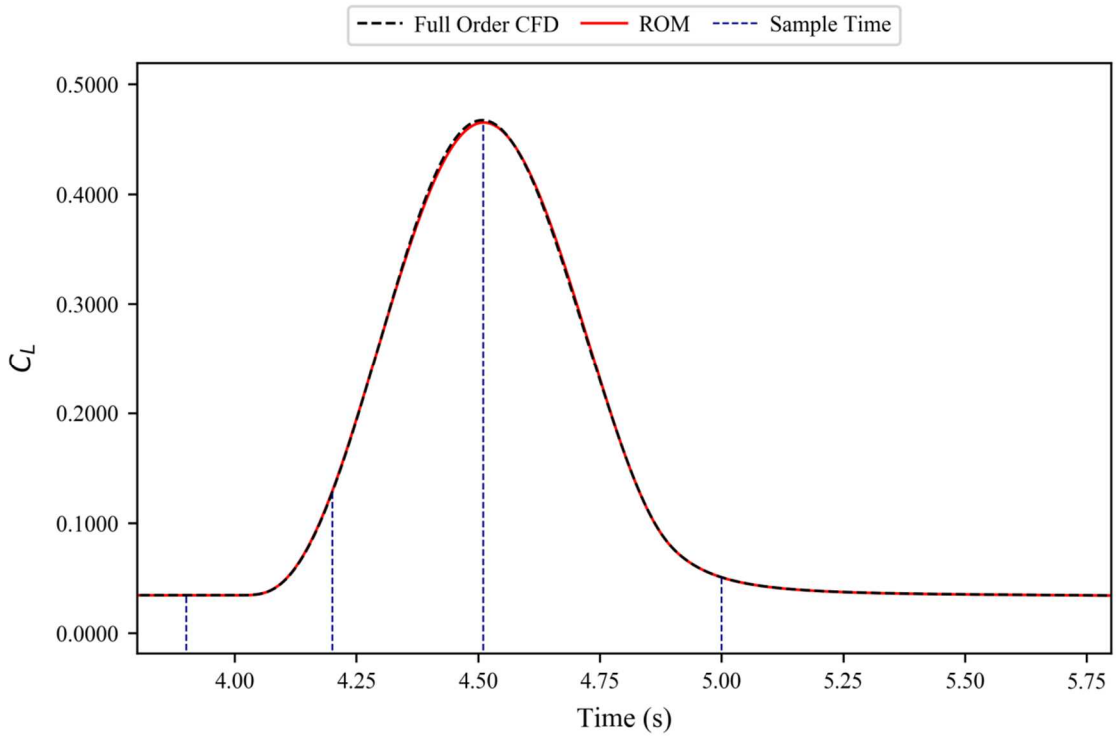

Figure 7.26. Locations of the sample times used for flight point 2, gust case 4.

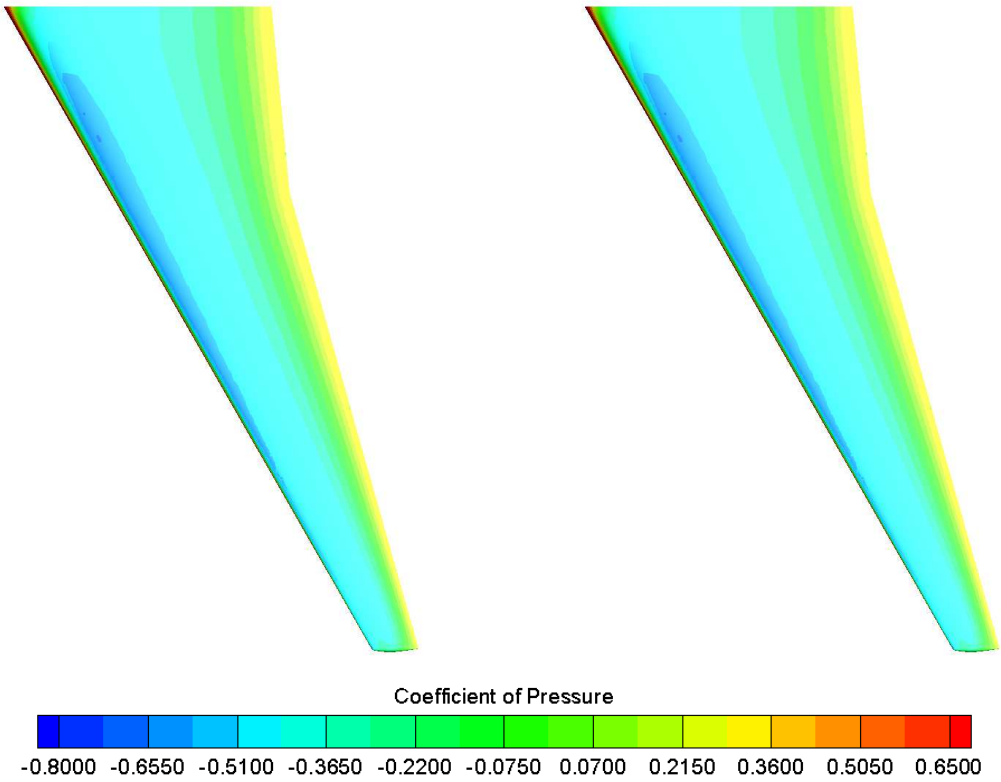

Figure 7.27. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 4 and at 3.900 seconds.

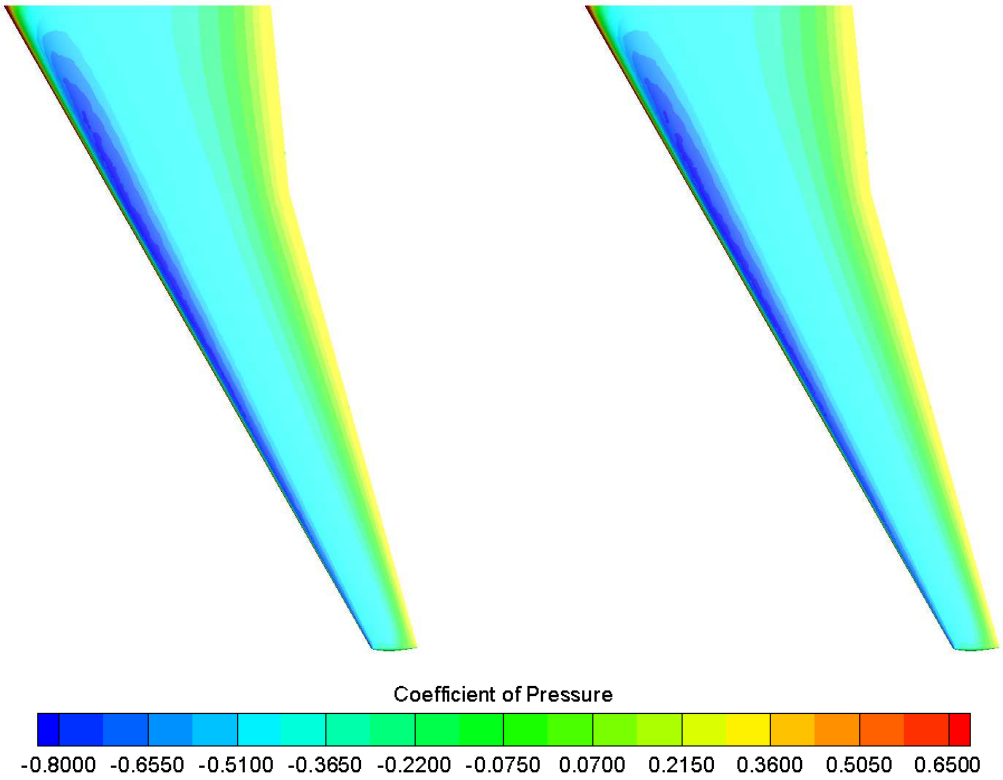

Figure 7.28. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 4 and at 4.200 seconds.

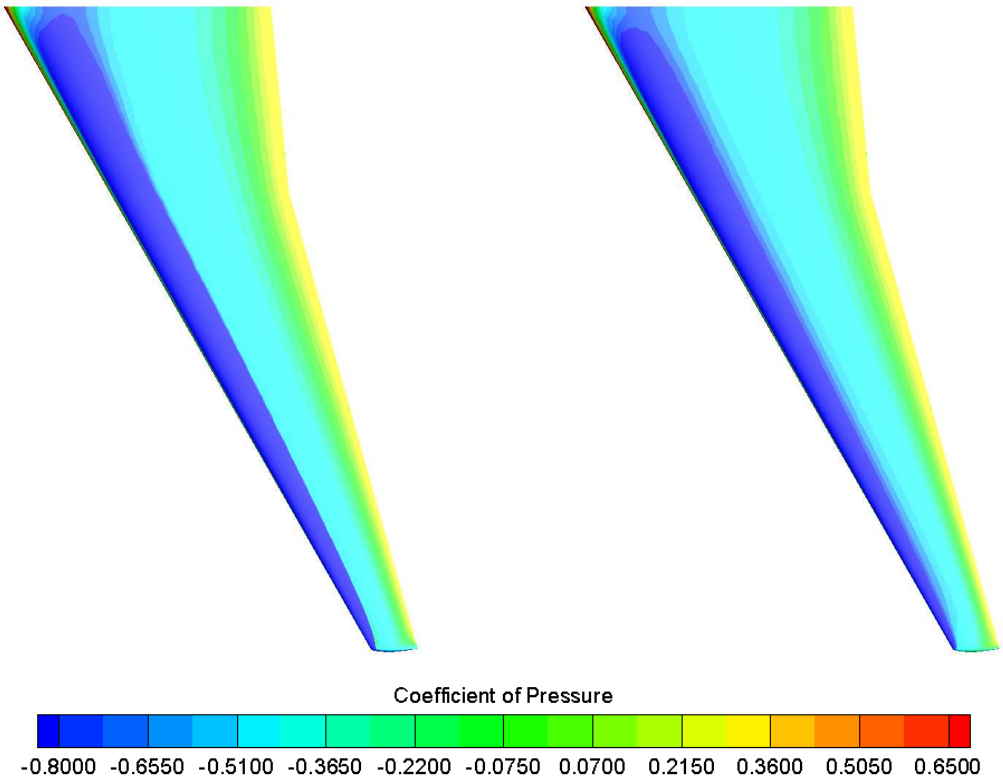

Figure 7.29. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 4 and at 4.510 seconds.

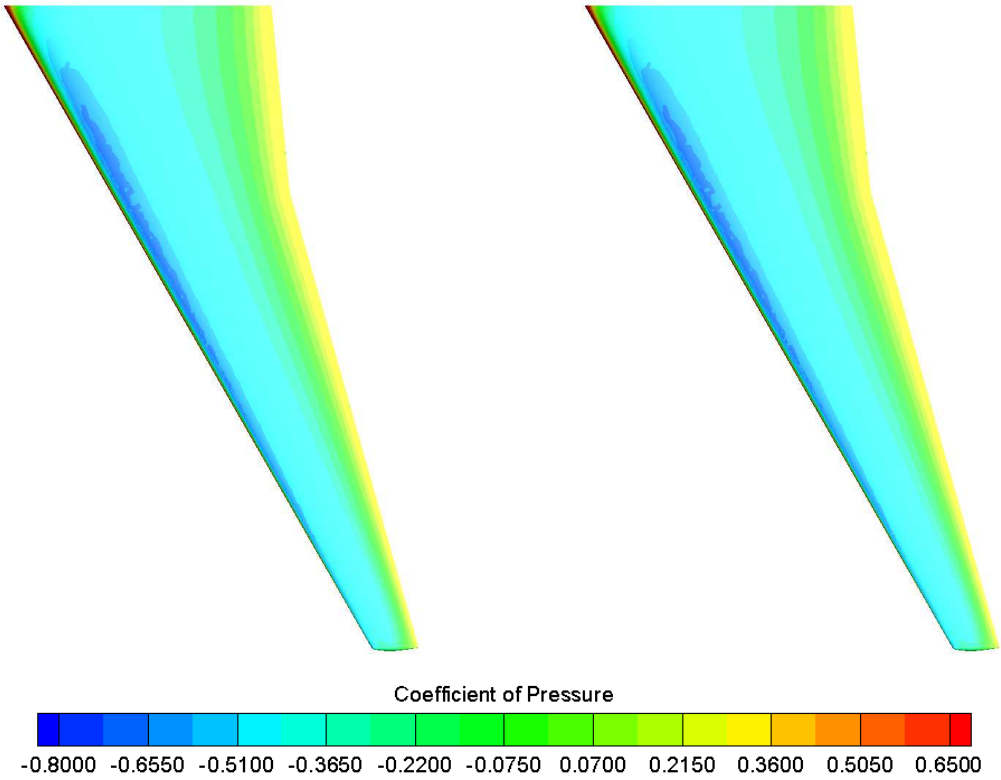

Figure 7.30. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the upper surface of the FFAST wing at flight point 2, gust case 4 and at 5.000 seconds.

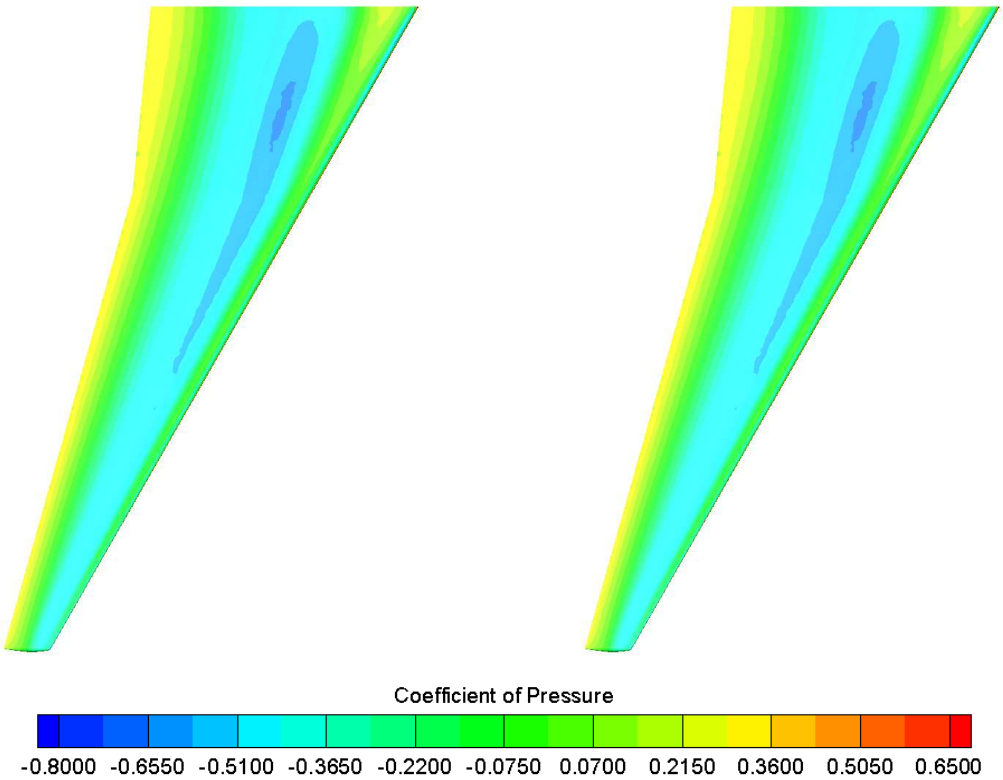

Figure 7.31. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 4 and at 3.900 seconds.

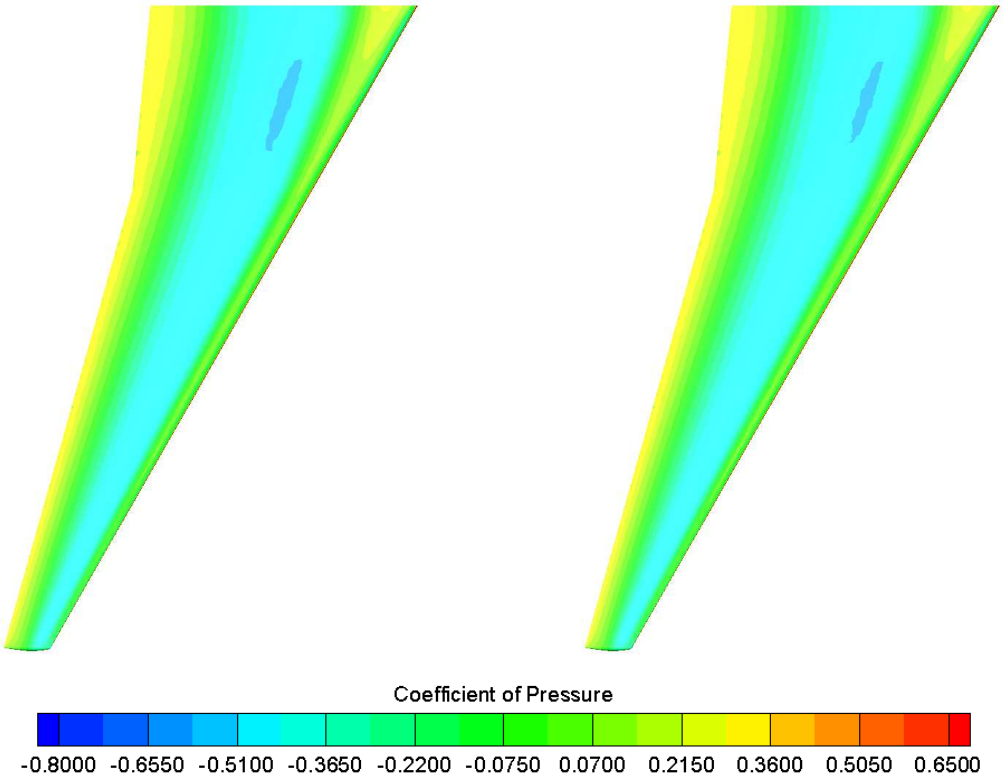

Figure 7.32. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 4 and at 4.200 seconds.

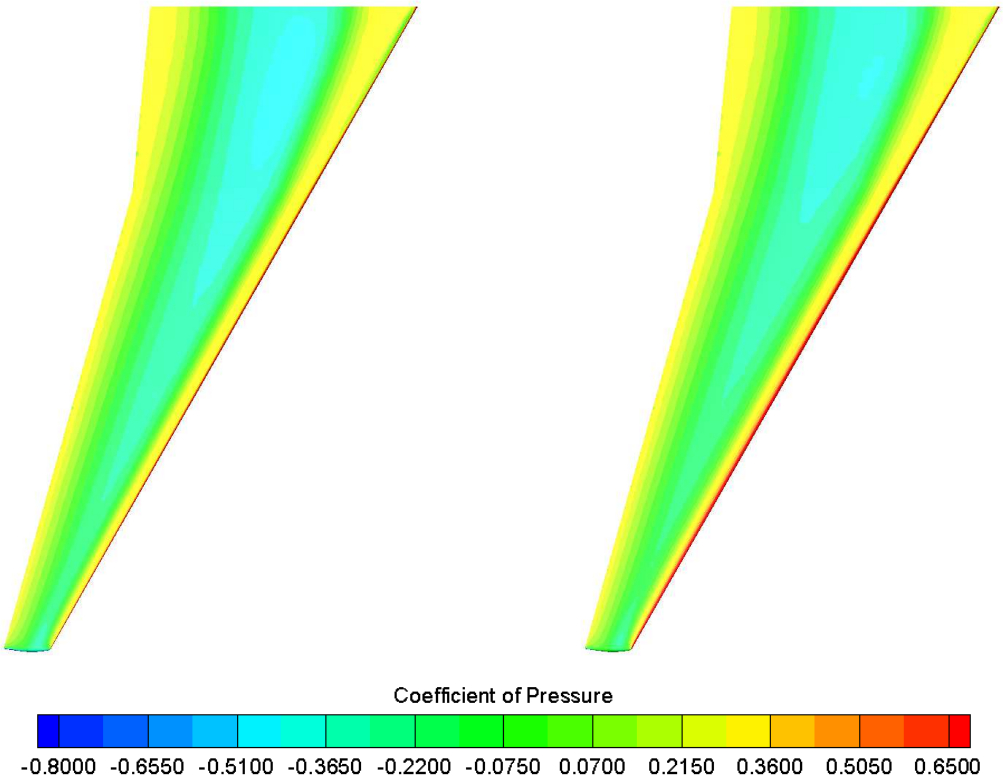

Figure 7.33. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 4 and at 4.510 seconds.

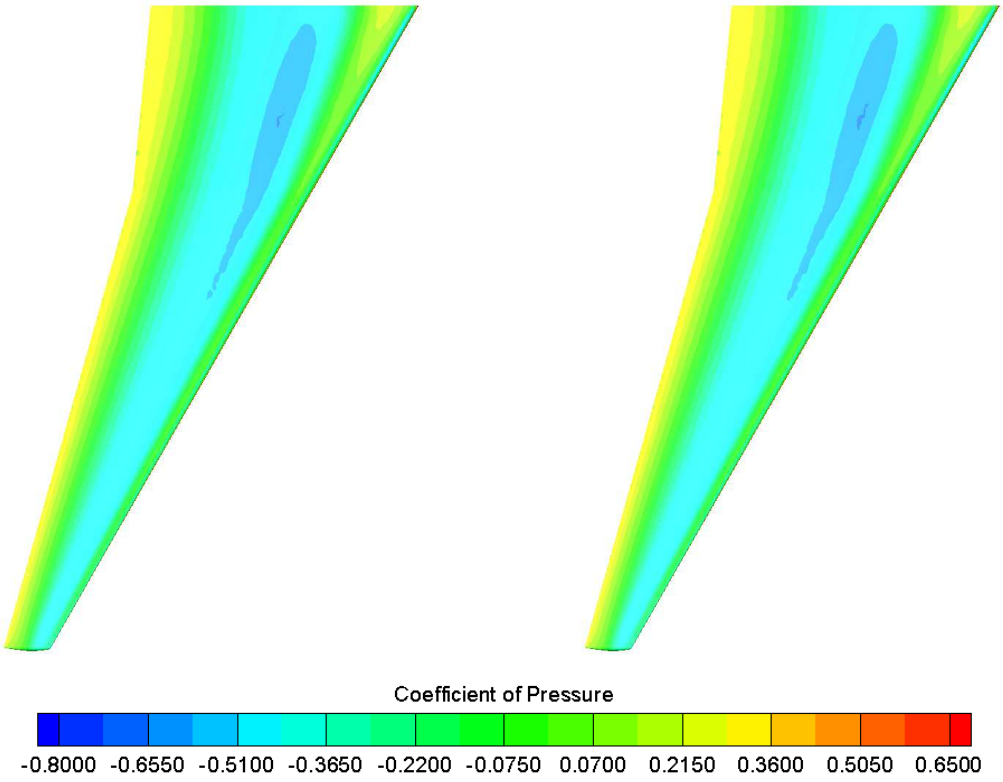

Figure 7.34. Comparison of CFD (left) and a pitching moment based ROM (right) obtained pressure coefficients for the lower surface of the FFAST wing at flight point 2, gust case 4 and at 5.000 seconds.

Figures 7.27-7.34 again show a general very strong match between the reconstructed surface pressures and those obtained via full order CFD. It is worth noting that at the peak (Figures 7.29 and 7.33) there is a very slight difference between the reconstructed and CFD surface pressures. However, this corresponds to the slight difference in the ROM results compared to the full order CFD ones at the peak of this gust length (see Figures 4.65 and 4.66) and is therefore an expected discrepancy between the two results.

7.2.2. Integrated Force Coefficient Results

Another way to compare the reconstructed surface pressures to those obtained via full order CFD is to integrate the surface pressures in strips across the model in order to obtain a force coefficient at each time step. From this, it is possible to compare the two sets of results in a similar manner as in other parts of this thesis.

Figures 7.35-7.42 show that, in general, the modified ROM was capable of producing a set of surface pressures which, when integrated, strongly matched the force coefficient

changes obtain via full order CFD. However, the figures also highlight a problem with the method.

It can be seen in each case that as soon as the gust being modelled gets to the time corresponding to the final sharp-edged gust surface pressure, the results rapidly degrade in accuracy. This would suggest a potential stability issue that is linked to the amount of sharp-edged gust surface pressure solutions that are used by the modified ROM. However, in order to fully establish this relationship, it is necessary to look at the outputs produced by the ROM when reading in sharp-edged gust surface pressures which end at different times. To help explore this, it is beneficial to use Flight Point 1; as its slower velocity (relative to Flight Point 2) means the model takes longer to pass through each gust.

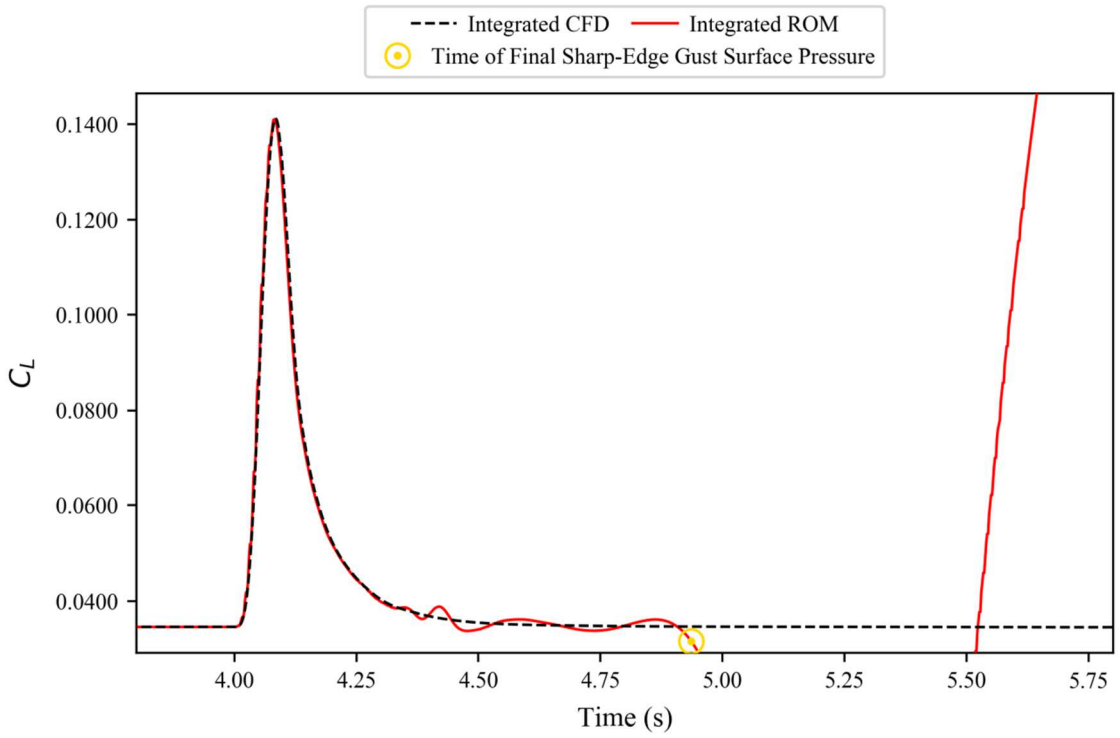

Figure 7.35. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 1.

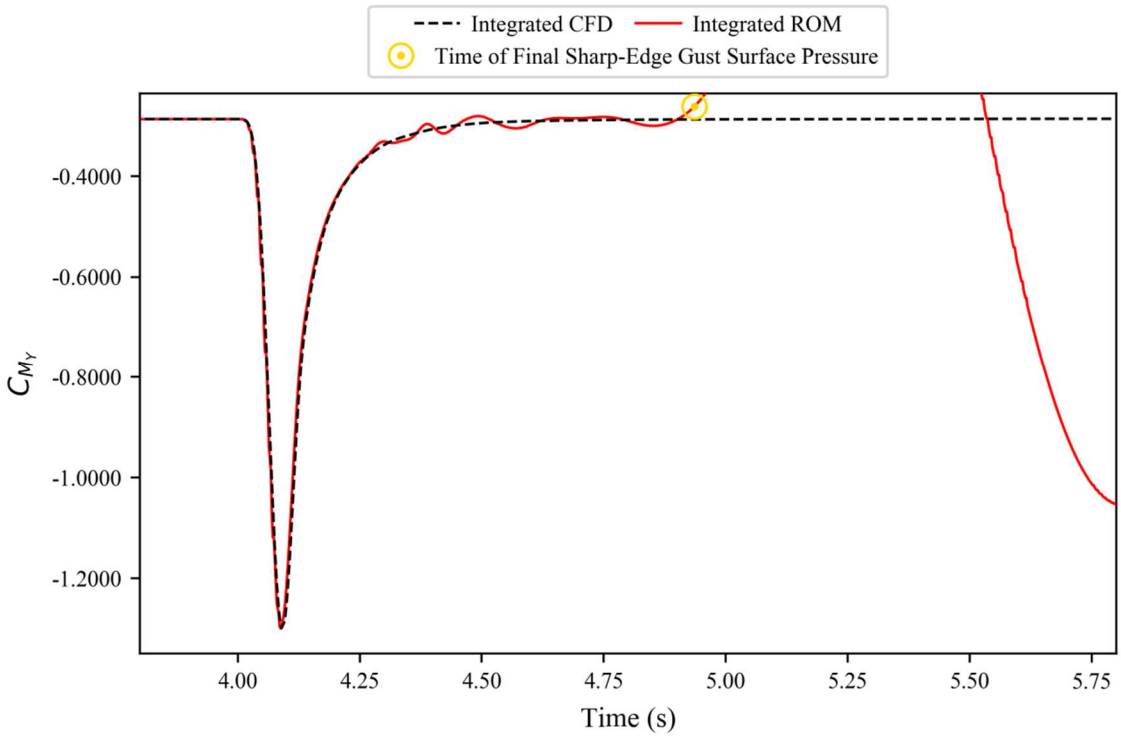

Figure 7.36. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 1.

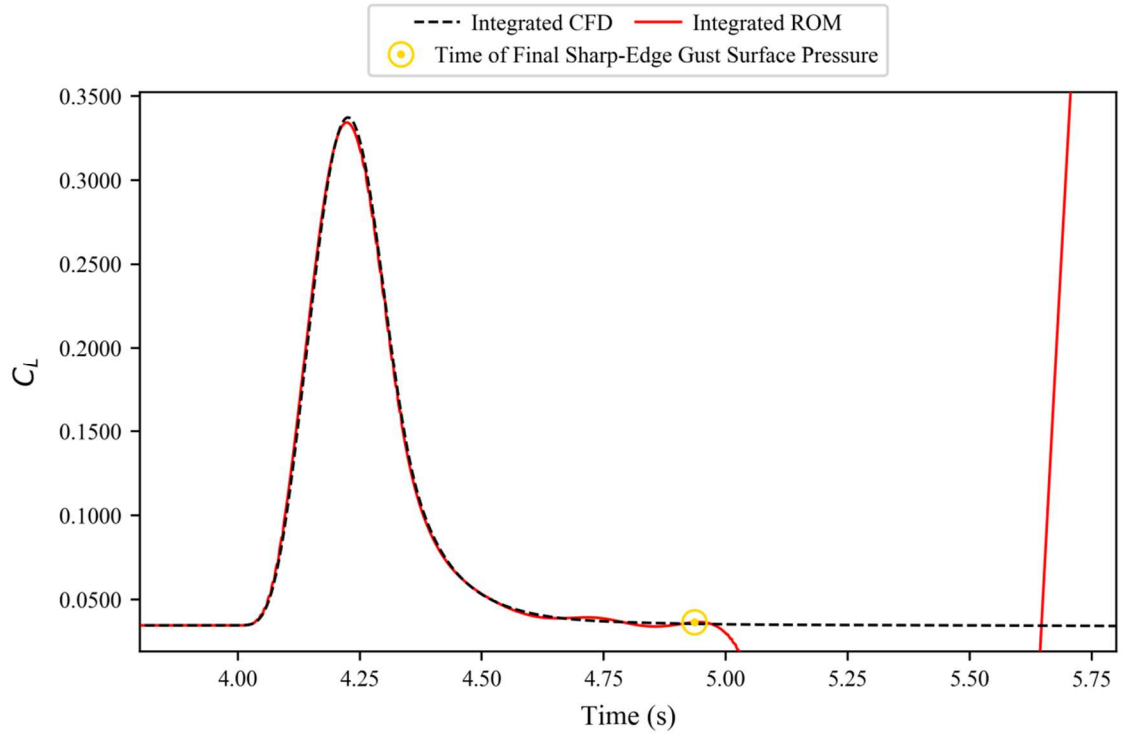

Figure 7.37. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 2.

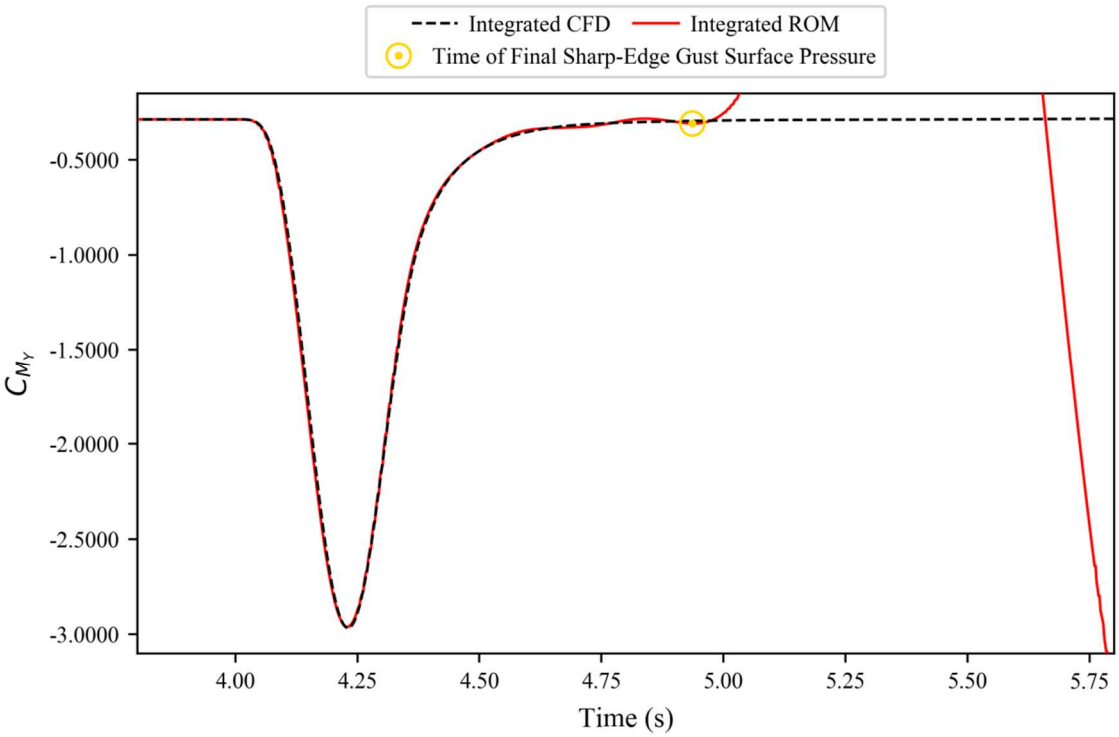

Figure 7.38. Coefficient of pitching, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 2.

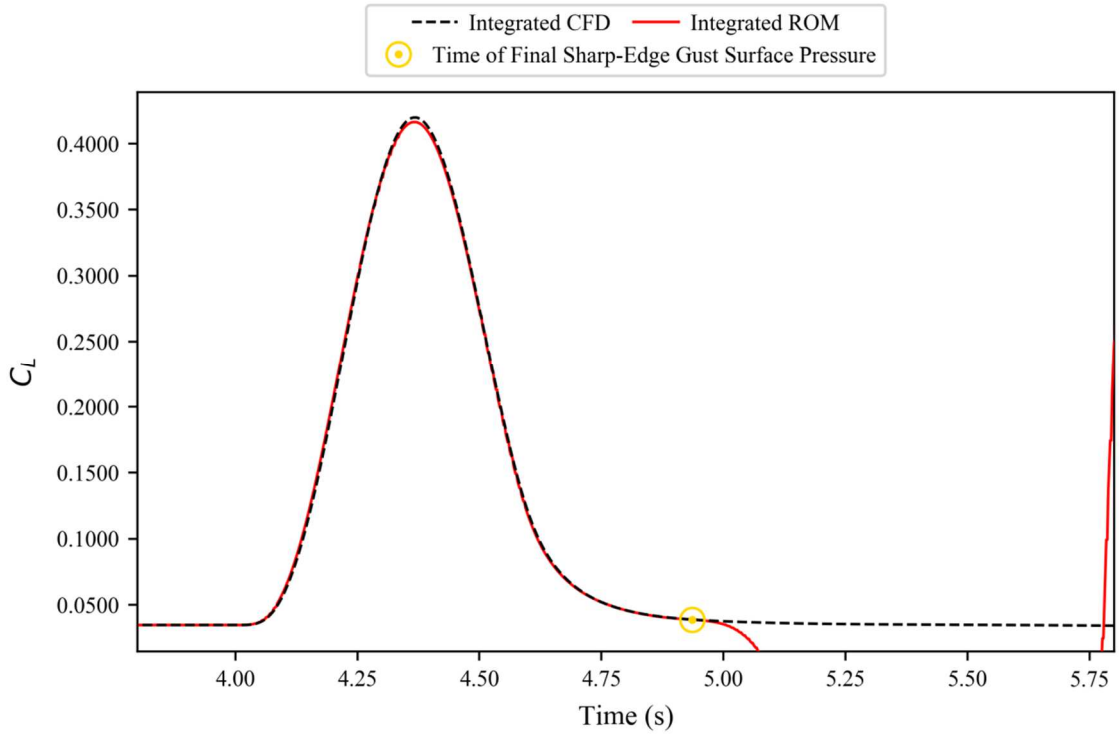

Figure 7.39. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 3.

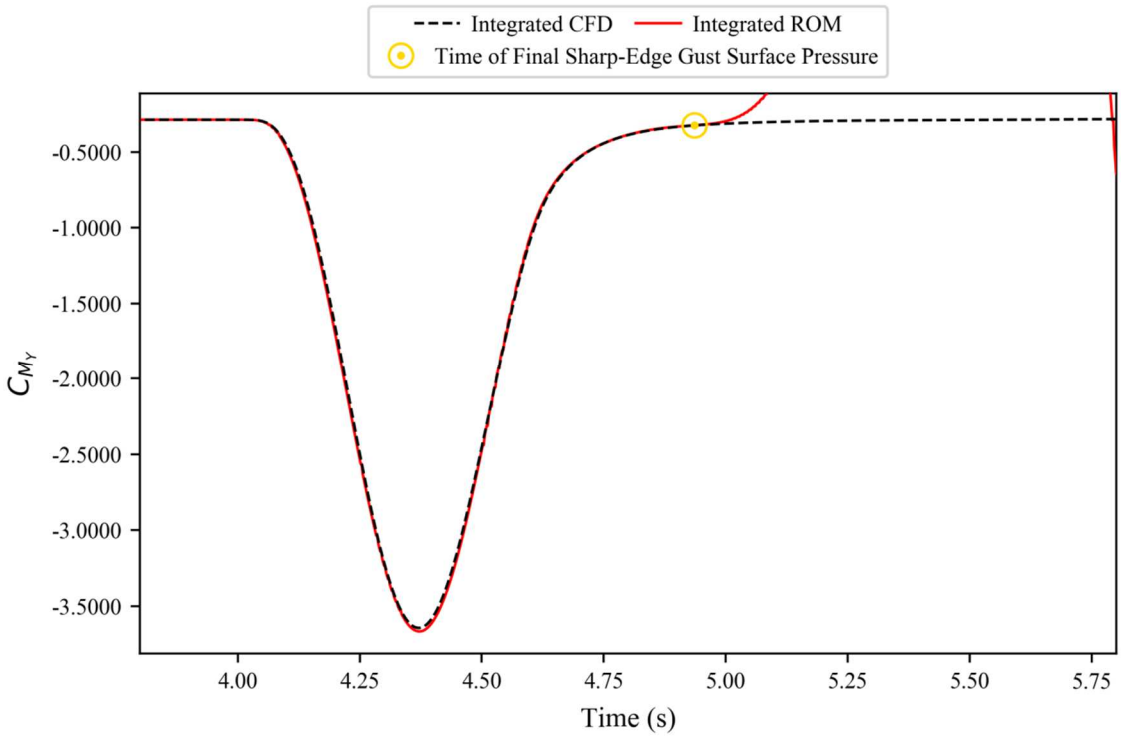

Figure 7.40. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 3.

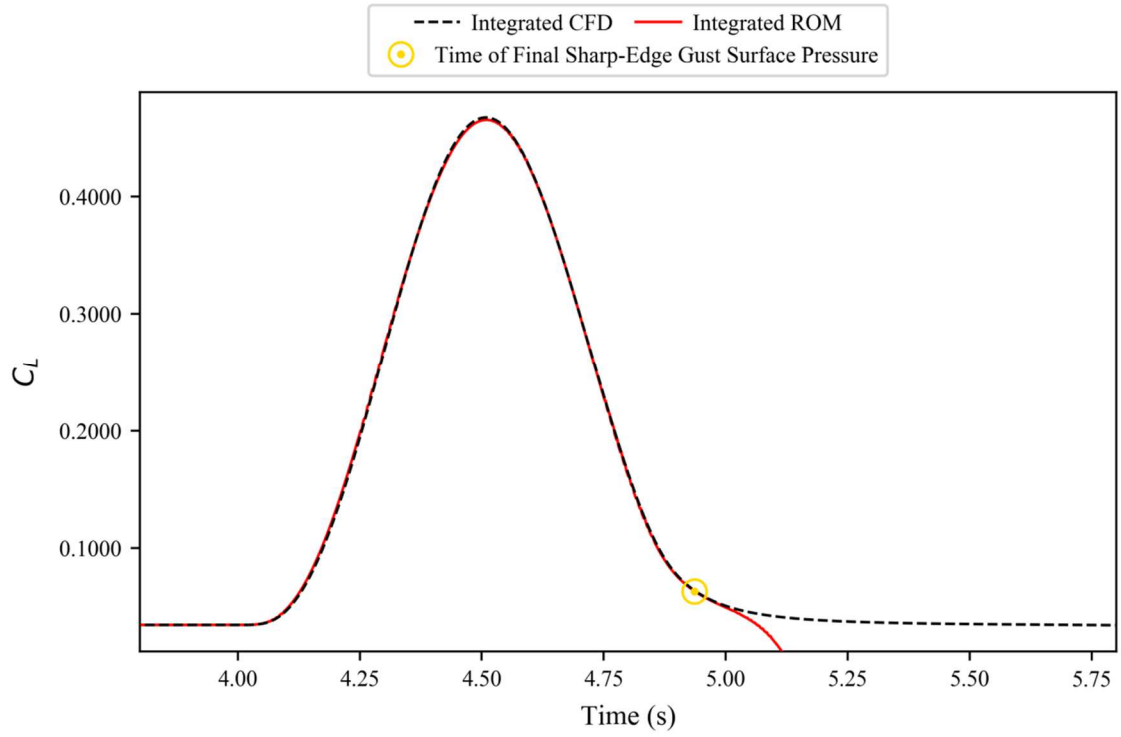

Figure 7.41. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 4.

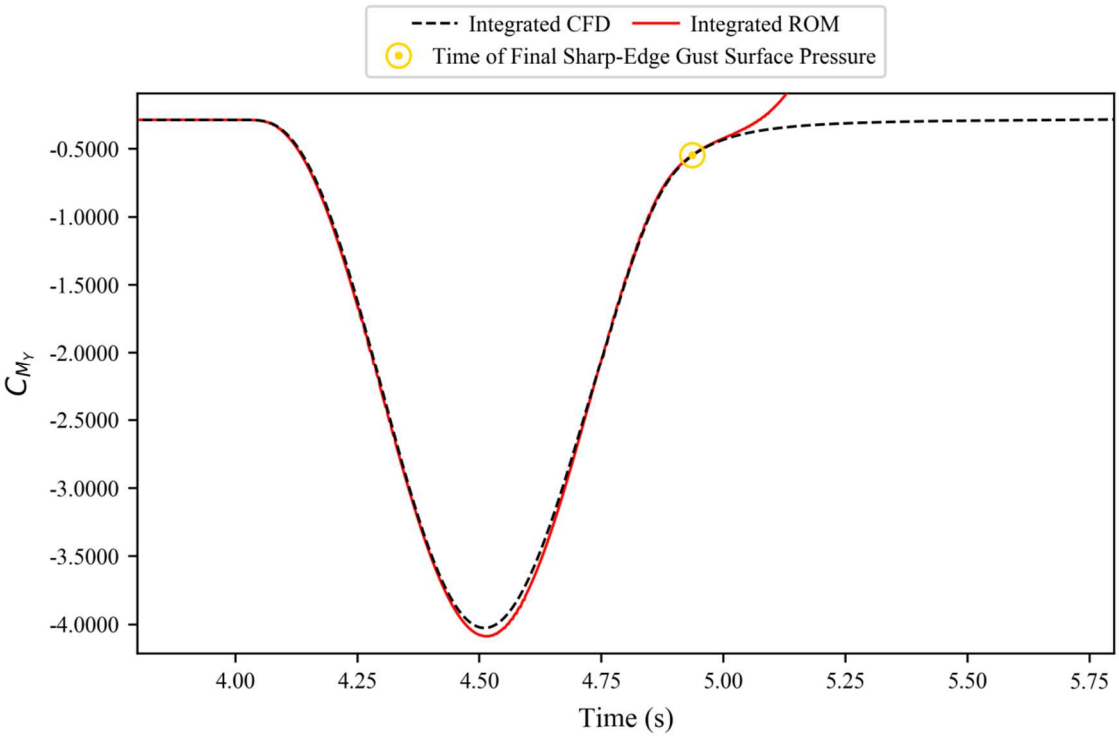

Figure 7.42. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 2, gust case 4.

It can be seen that for both a short gust (Figures 7.43-7.54) and a long gust (Figures 7.55-7.66) there is a clear trend. The surface pressure reconstruction starts with a high level of accuracy that rapidly decreases (often to the point of being simply wrong) when the time of the output gust exceeds the time of the final sharp-edged gust surface pressure used. In instances where this occurs before the peak of the response (as was the case in the long gust length) the output will typically maintain a good level of accuracy for a short time after this critical time; before the response dies out before the peak. In instances where this occurs after the peak, the response typically degrades extremely close to this critical time.

Additionally, there are three further trends that appear to be important and therefore worth highlighting. The first is that the earlier the critical time is the more extreme the degradation in accuracy appears to be. Secondly, if the degradation is sufficiently large, and it occurs after the peak response, it can be seen to propagate back in time. Finally, the response can be seen returning to the steady state if run for a suitably long period after the gust (which can be seen in the short gust case of Figures 7.43-7.54). Together these observations would suggest that the issue is a new type of stability issue; not dissimilar to those discussed previously in this thesis. However, to conclusively define

(and then solve) the problem, further research would have to be undertaken; and due to time limitations this was not possible within the undertaking of this PhD.

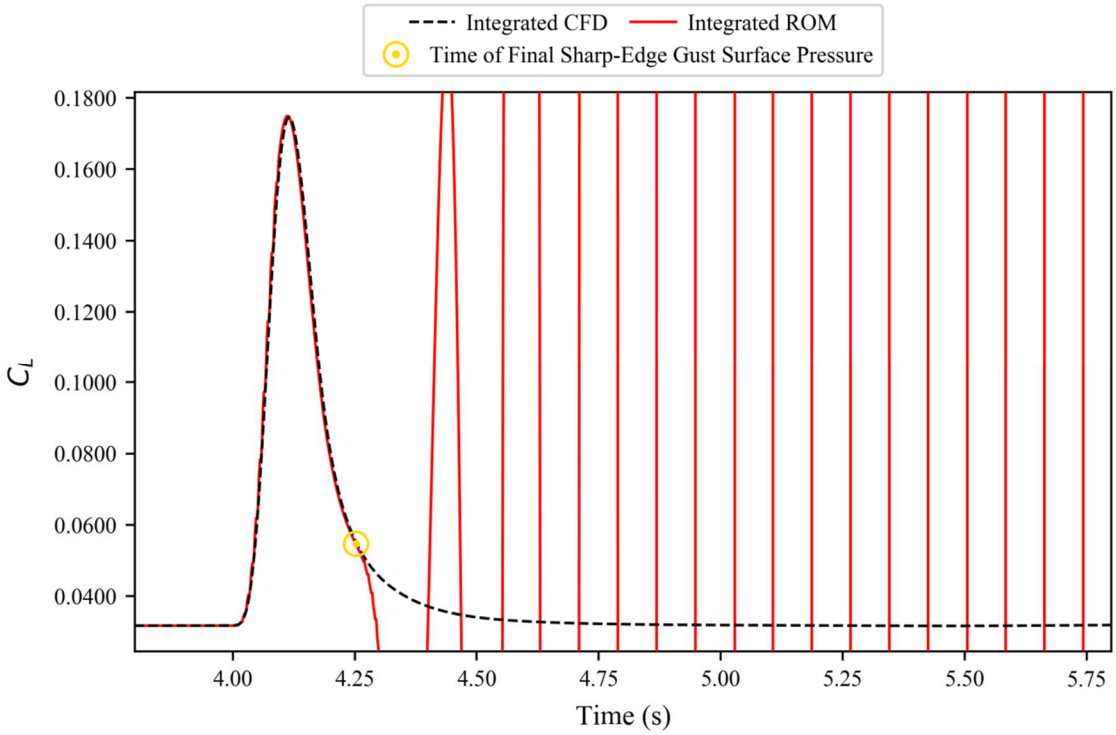

Figure 7.43. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.252 seconds.

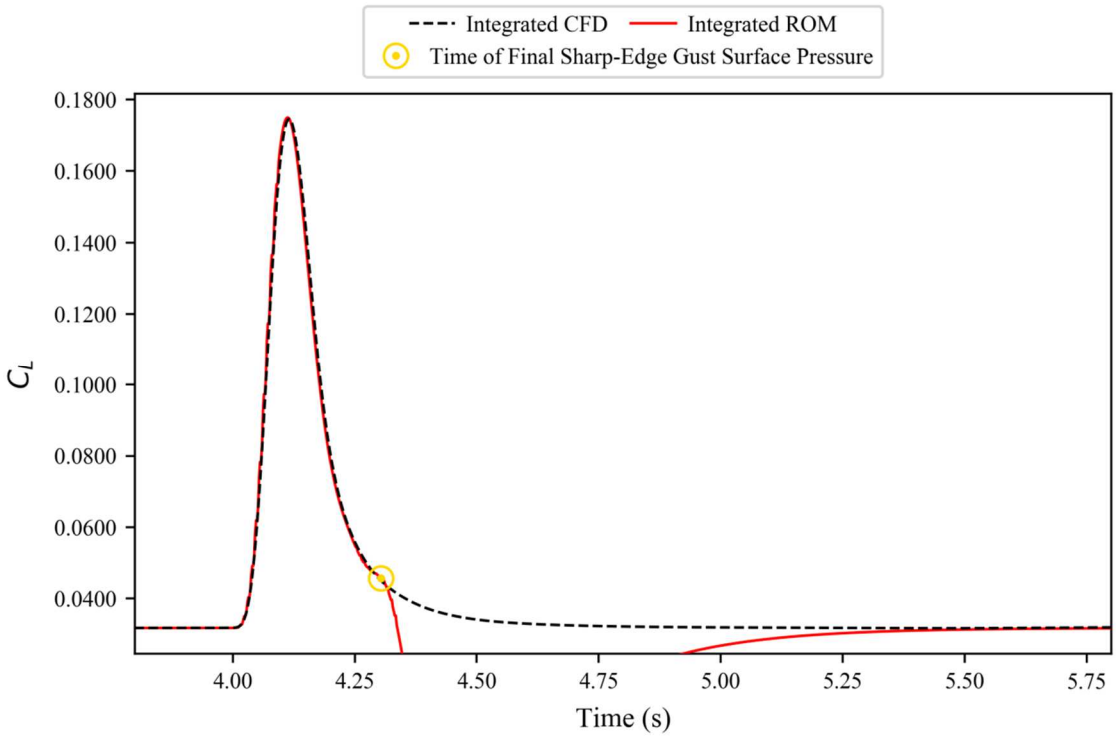

Figure 7.44. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.304 seconds.

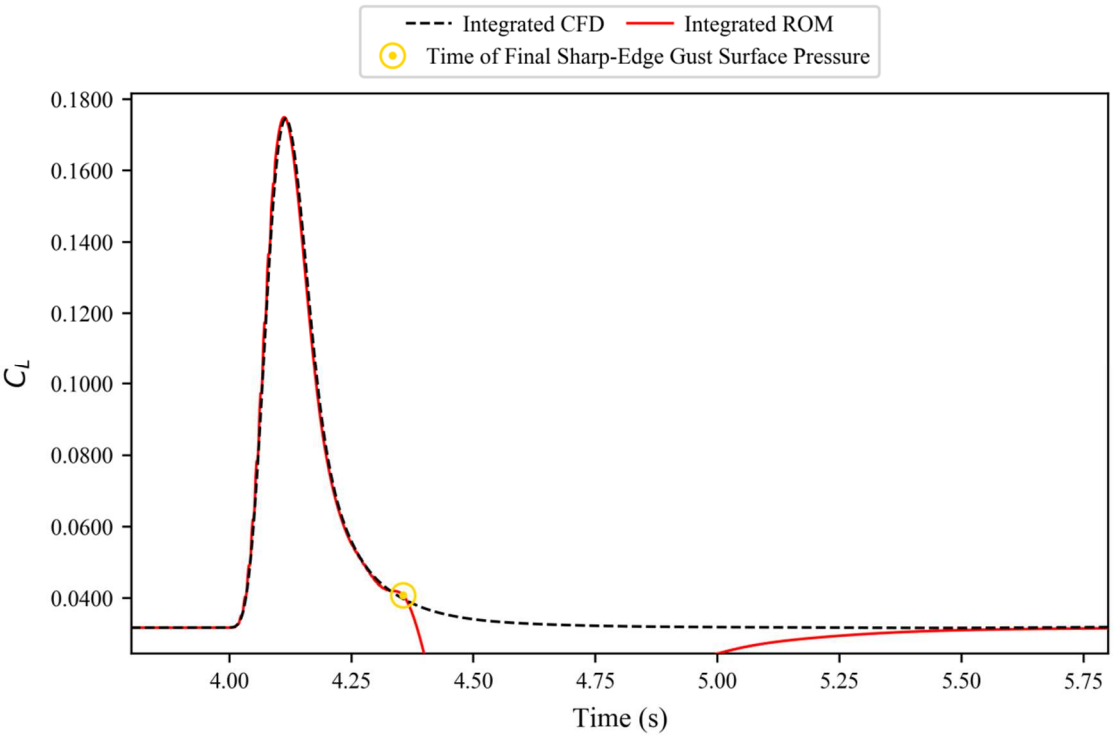

Figure 7.45. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.356 seconds.

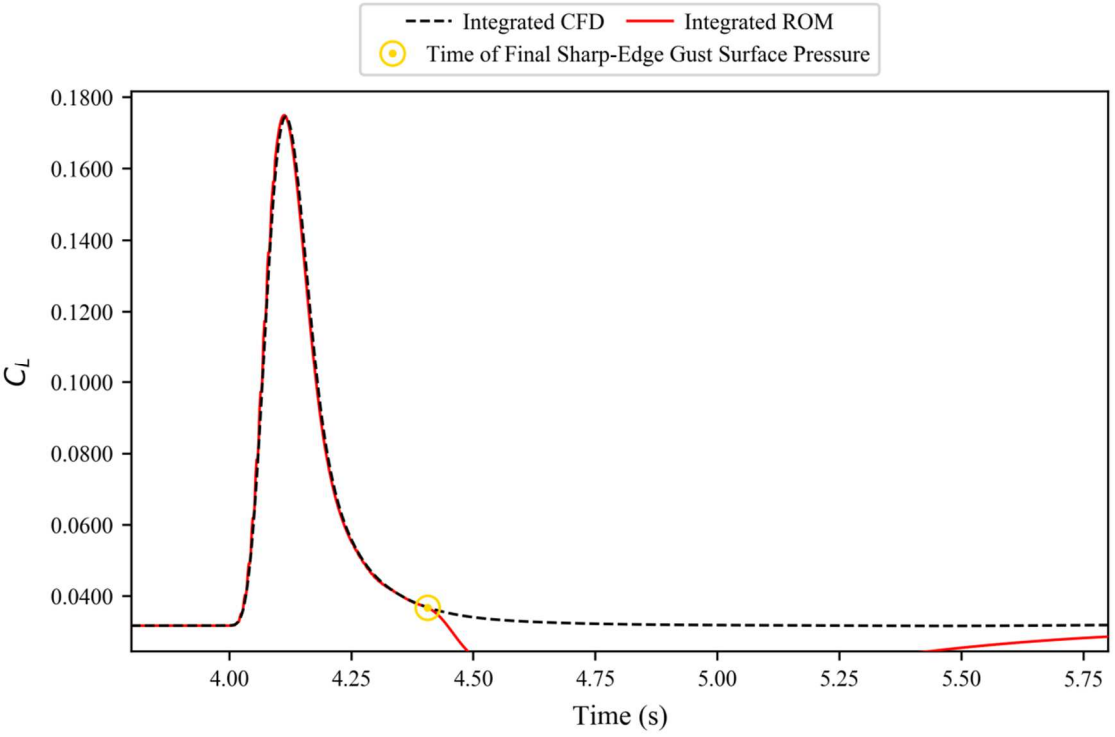

Figure 7.46. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.406 seconds.

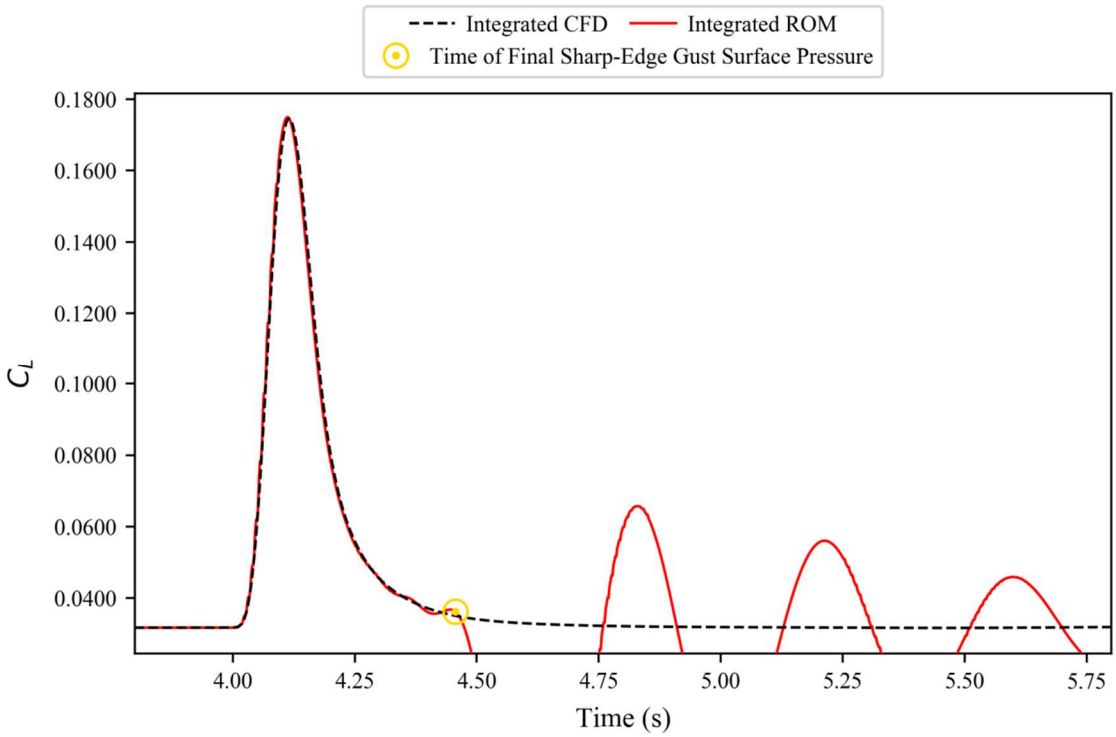

Figure 7.47. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.456 seconds.

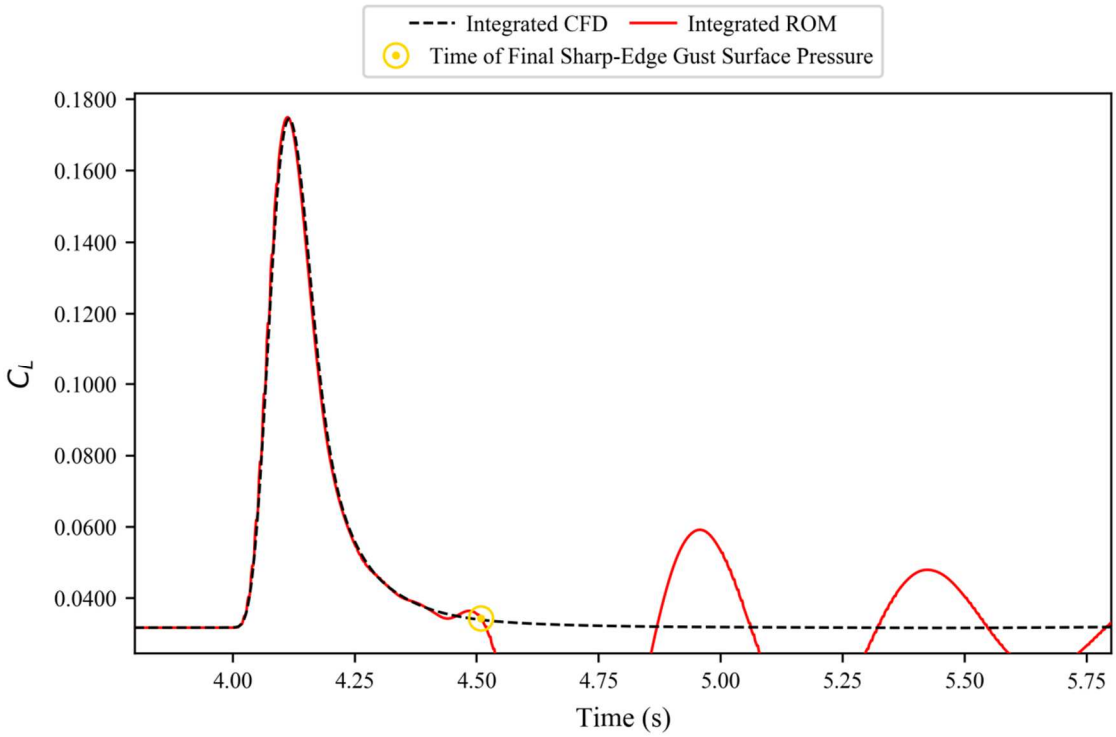

Figure 7.48. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.508 seconds.

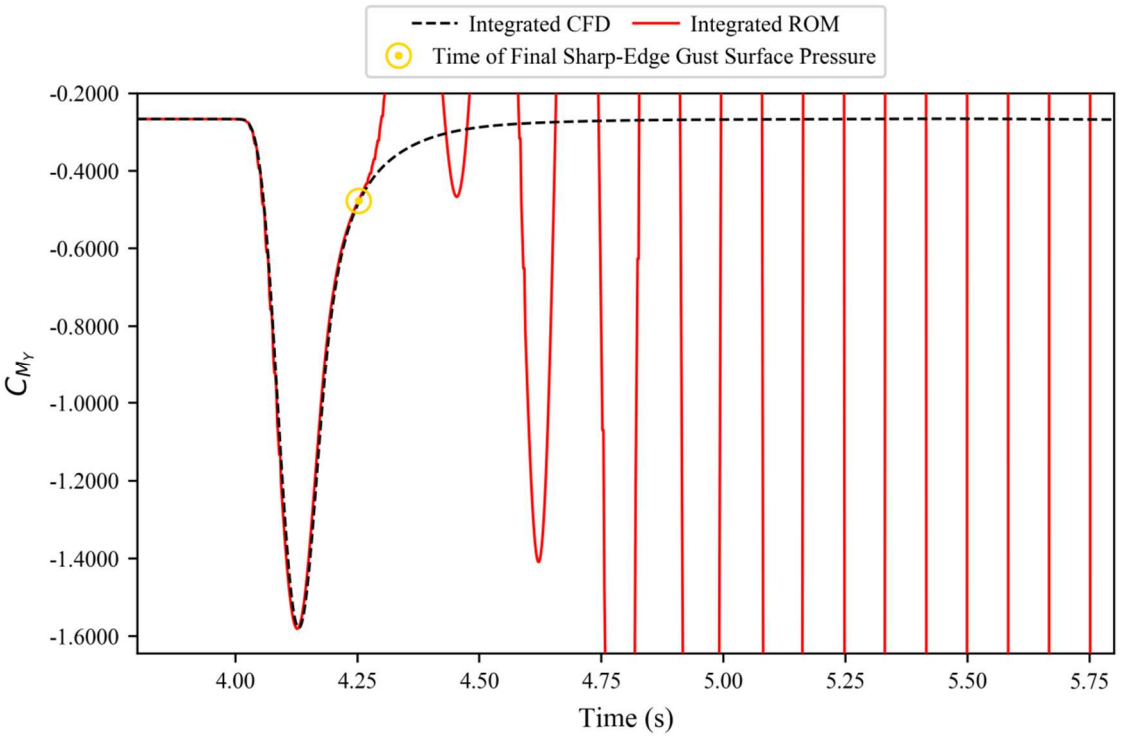

Figure 7.49. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.252 seconds.

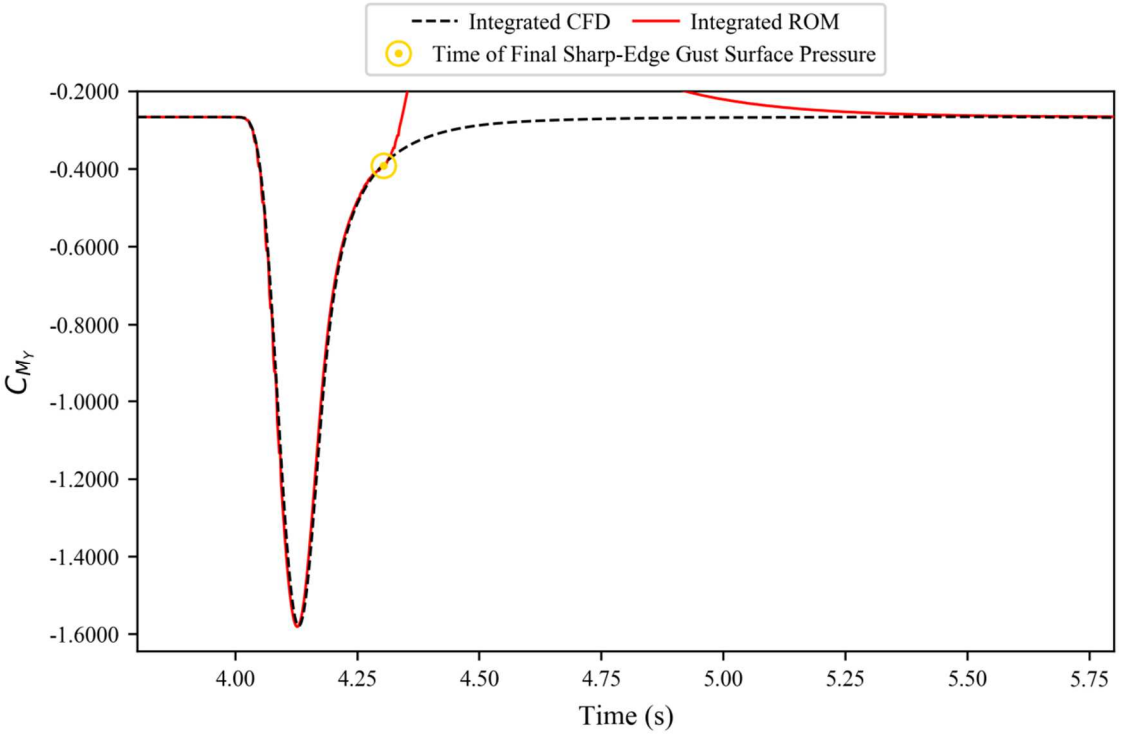

Figure 7.50. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.304 seconds.

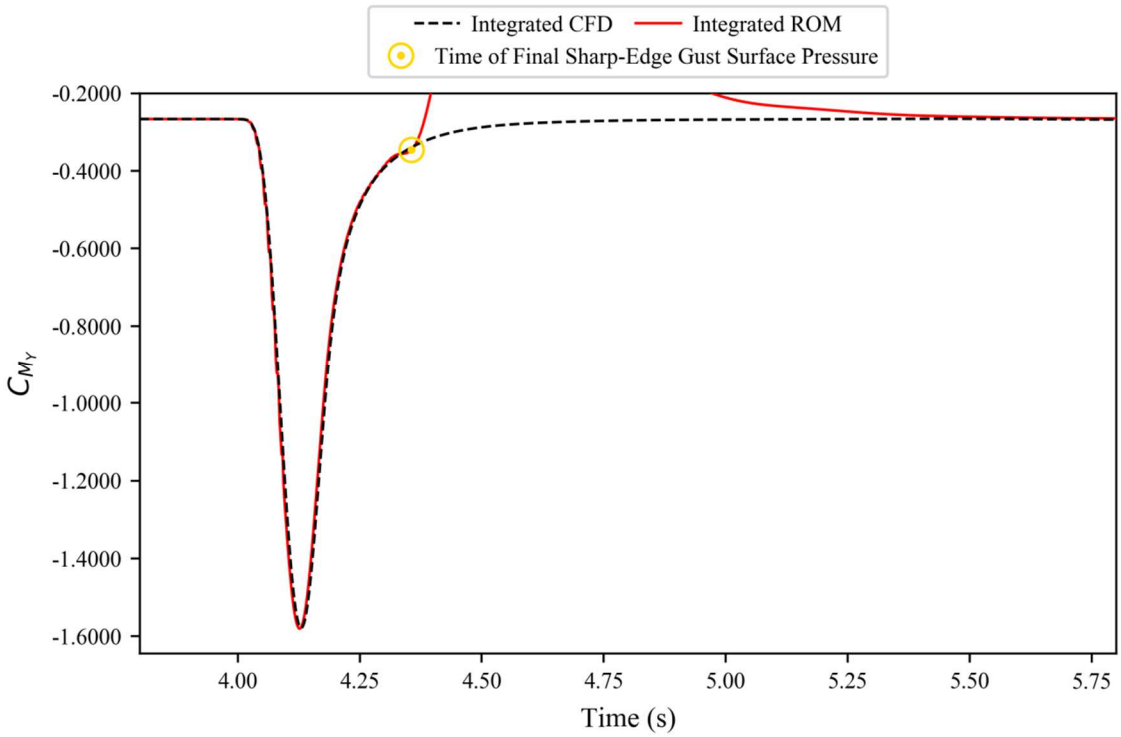

Figure 7.51. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.356 seconds.

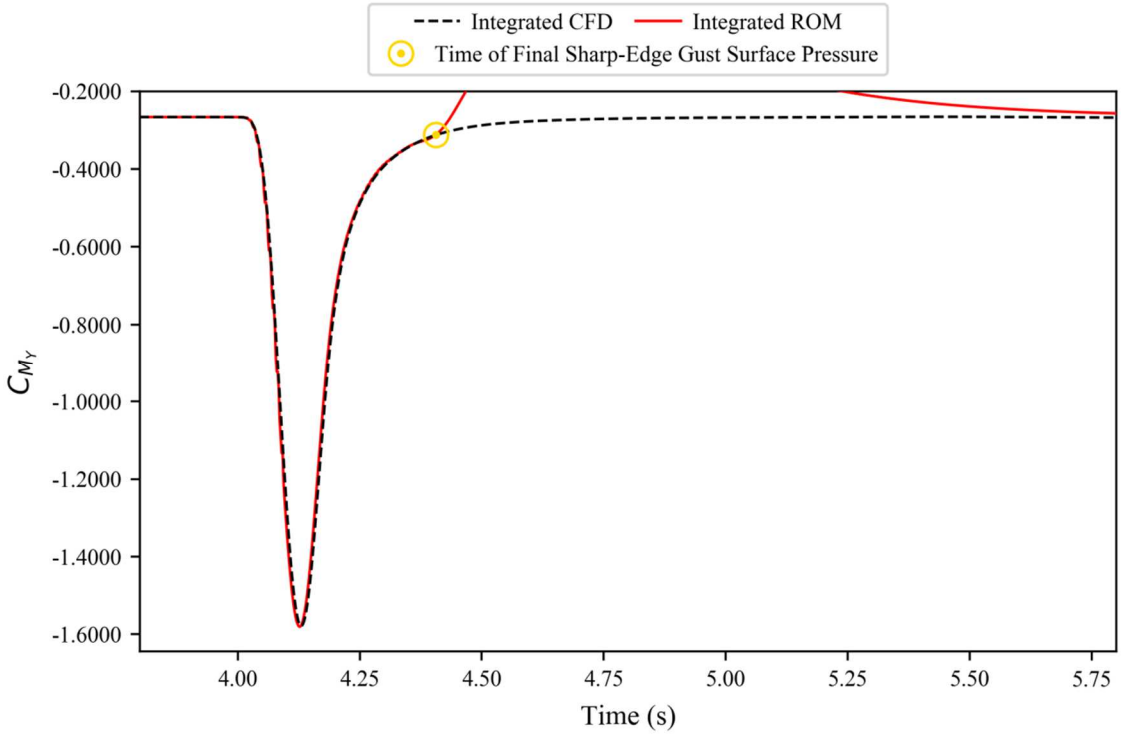

Figure 7.52. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.406 seconds.

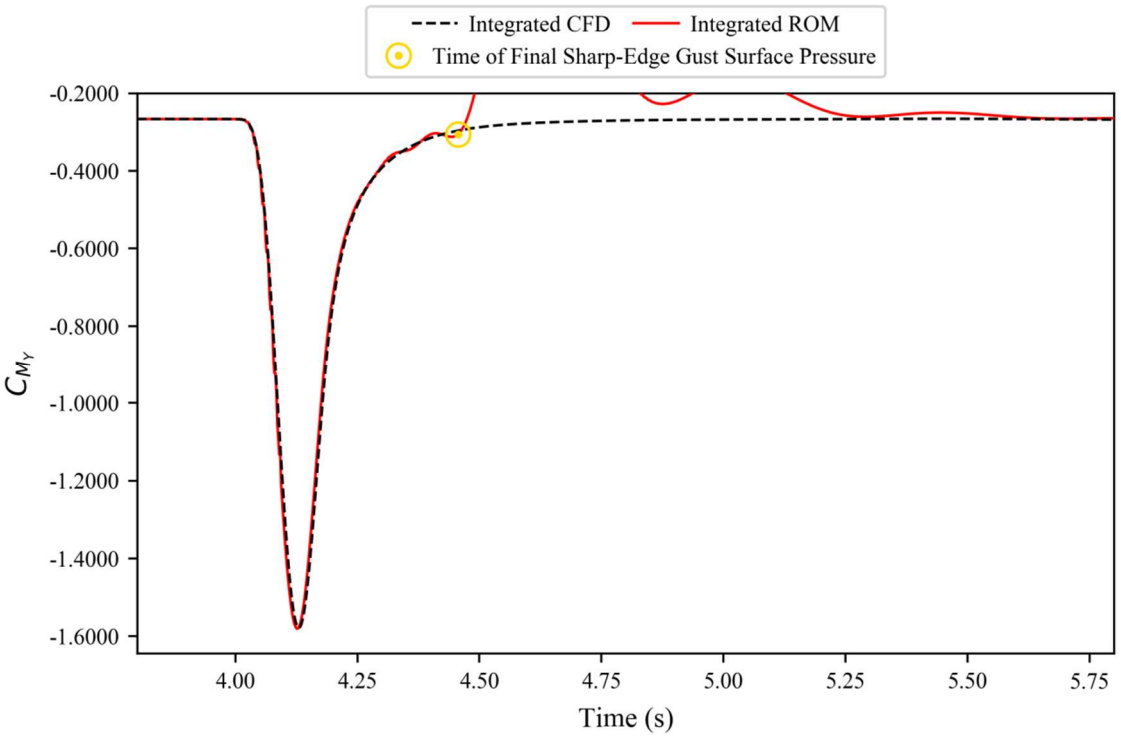

Figure 7.53. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.456 seconds.

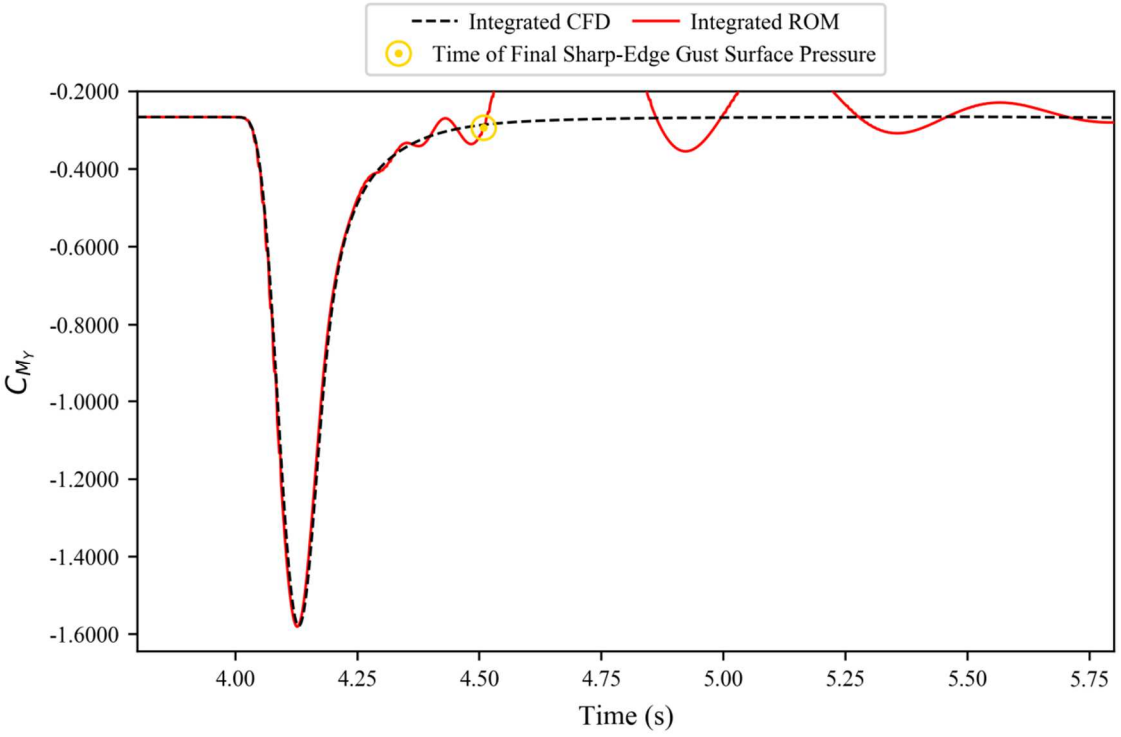

Figure 7.54. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 1. Created with a final, sharp-edged gust, surface pressure solution at 4.508 seconds.

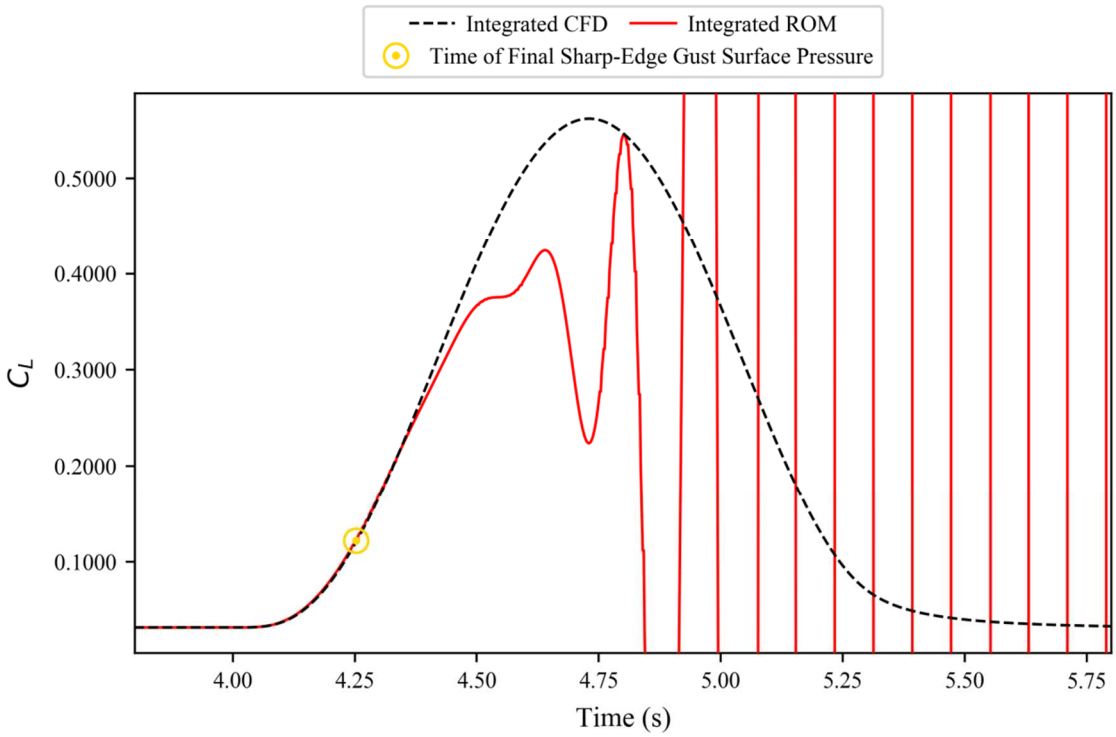

Figure 7.55. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.252 seconds.

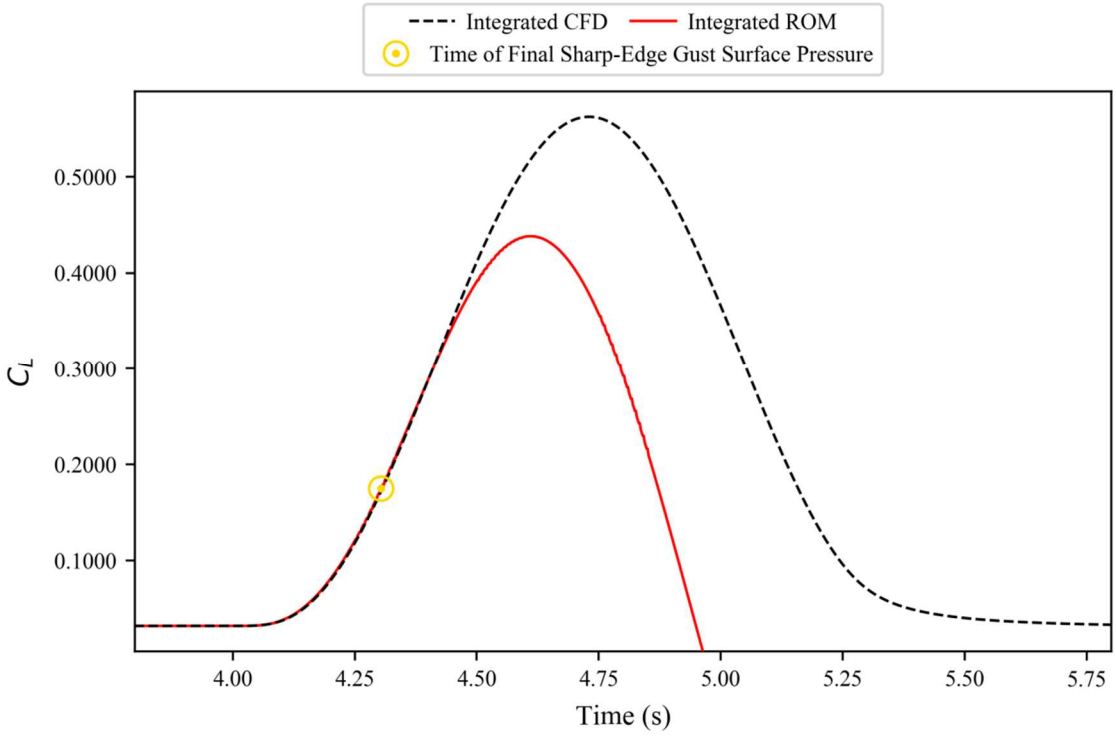

Figure 7.56. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.304 seconds.

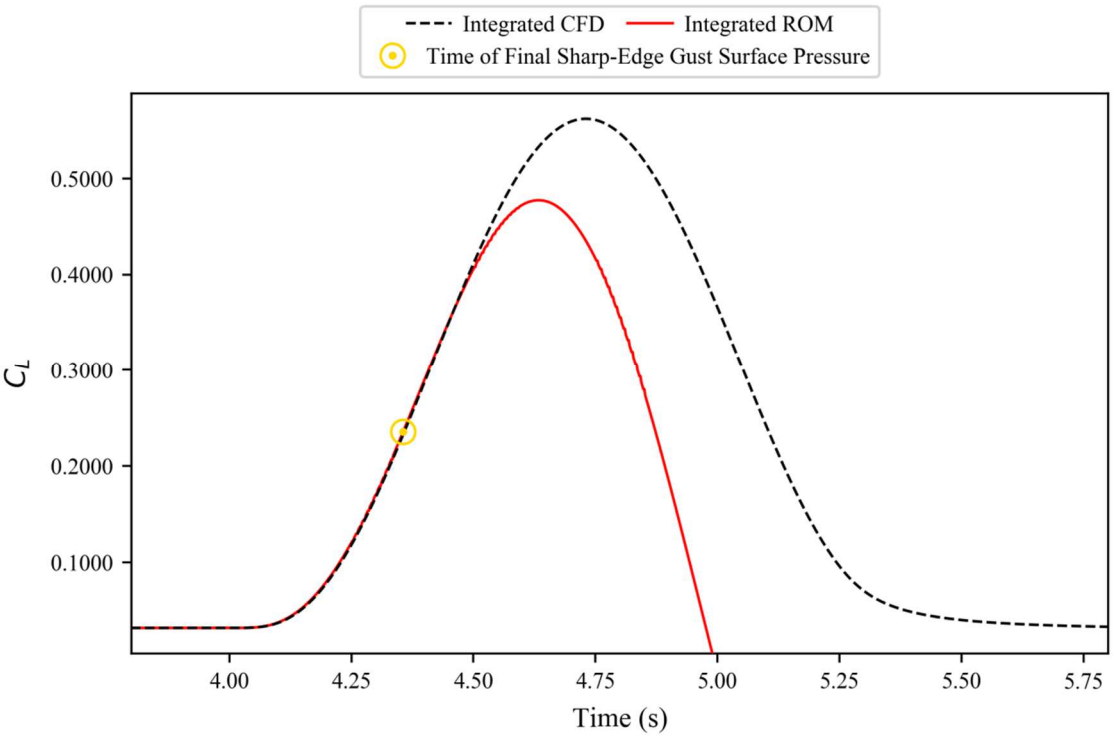

Figure 7.57. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.356 seconds.

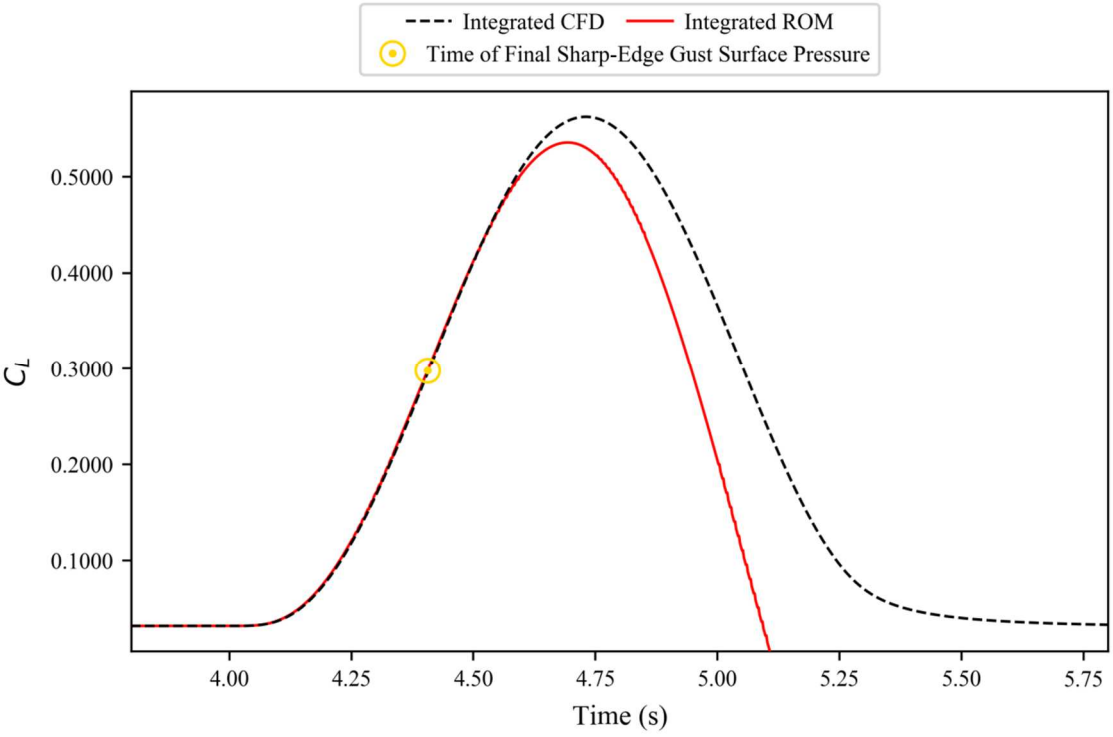

Figure 7.58. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.406 seconds.

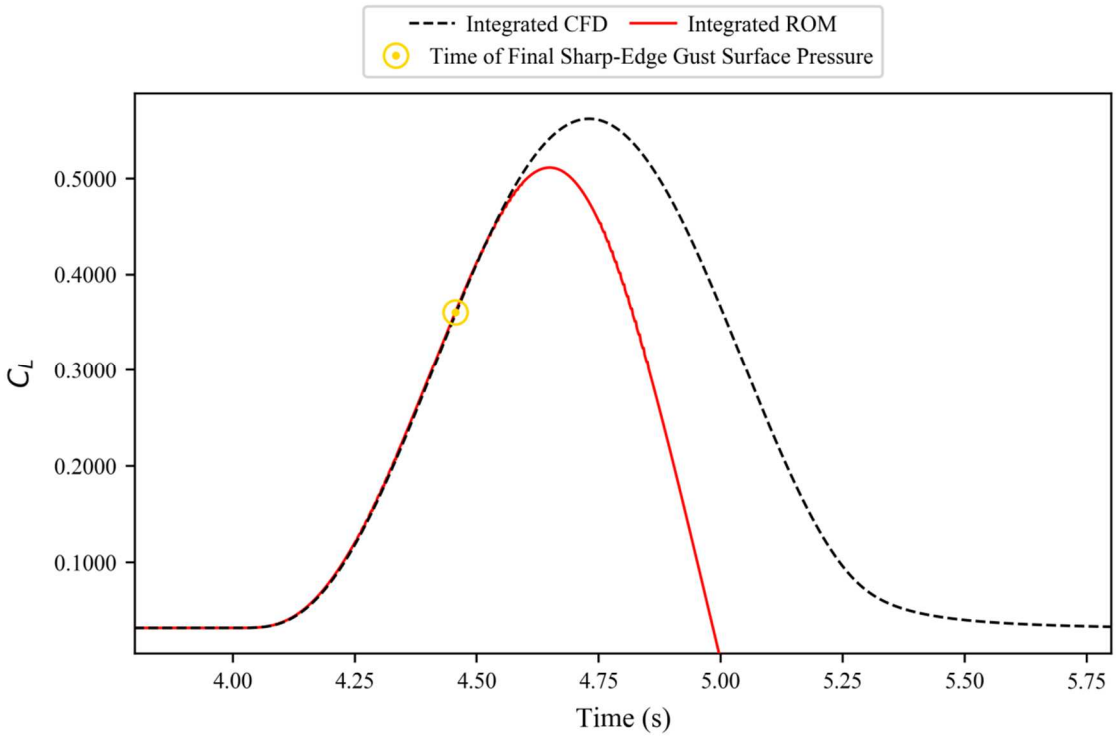

Figure 7.59. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.456 seconds.

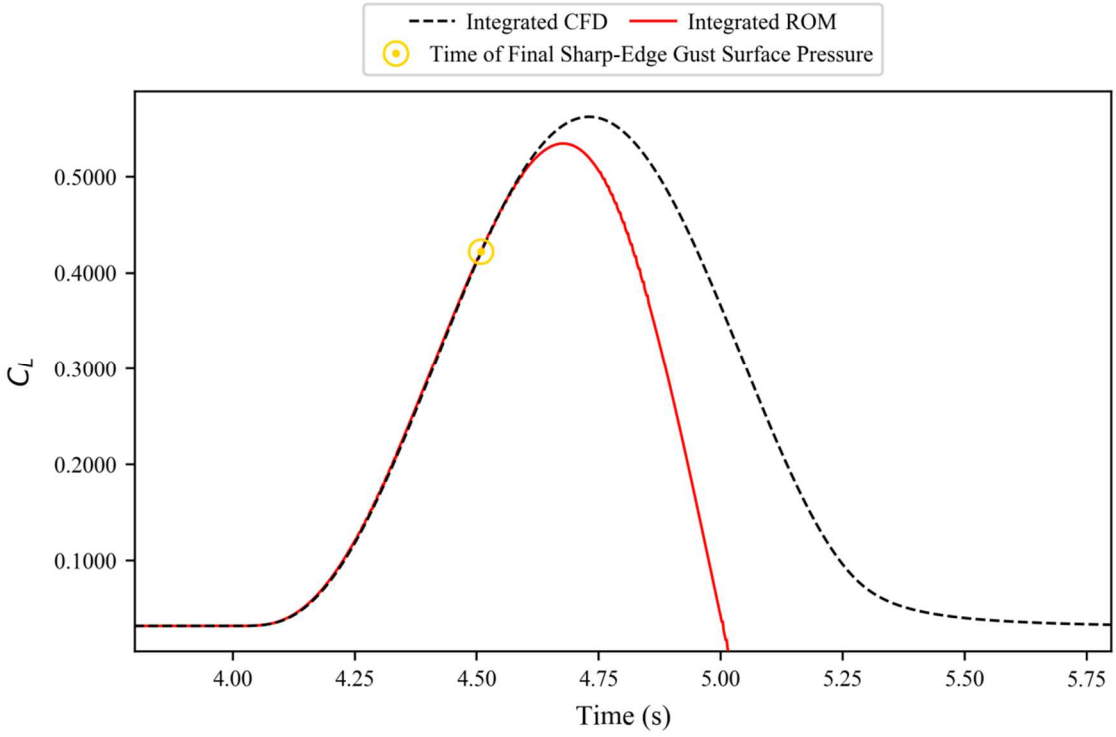

Figure 7.60. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.508 seconds.

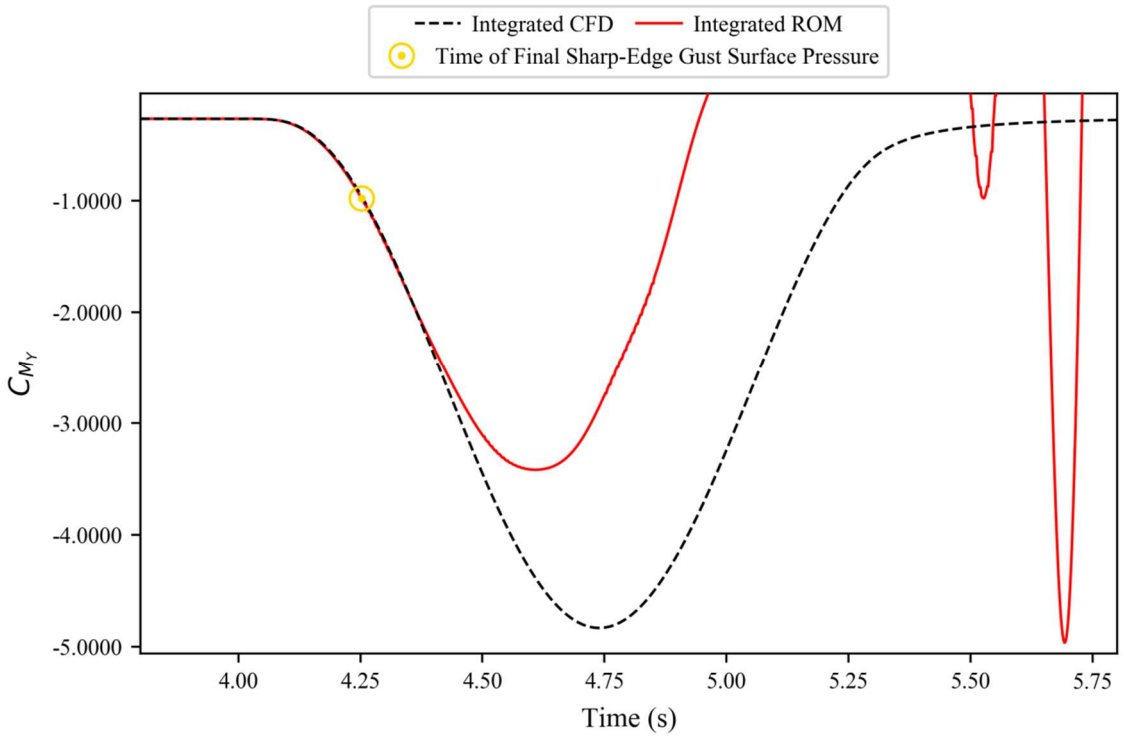

Figure 7.61. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.252 seconds.

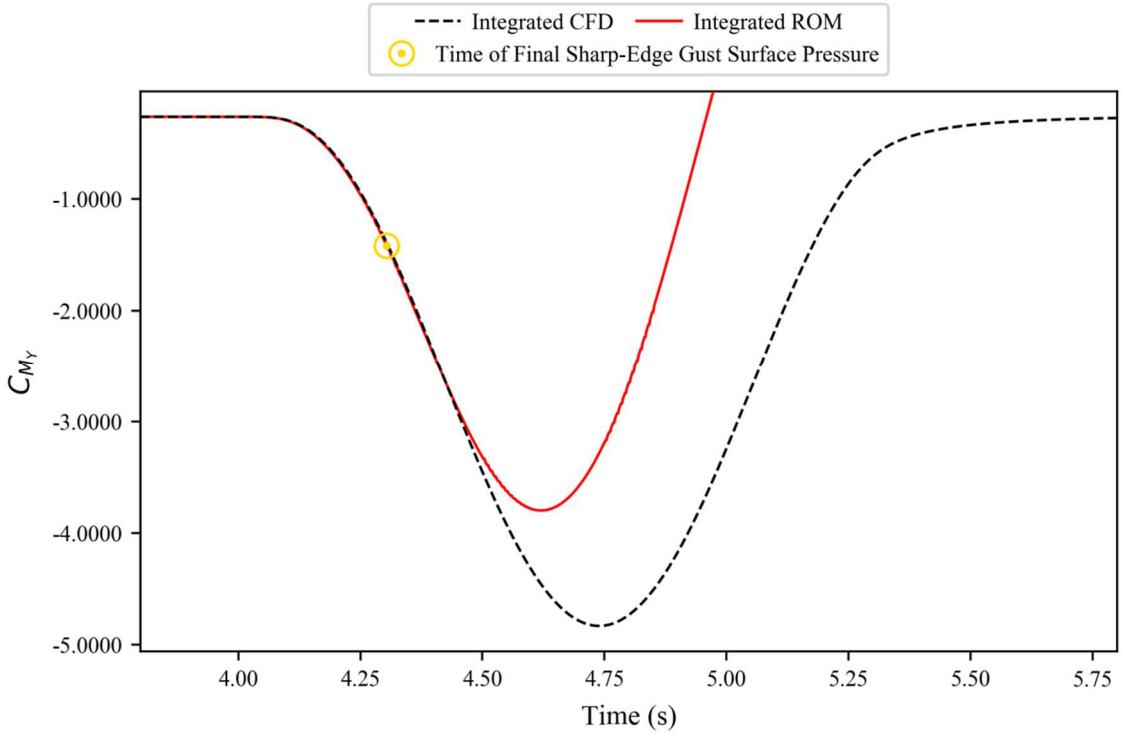

Figure 7.62. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.304 seconds.

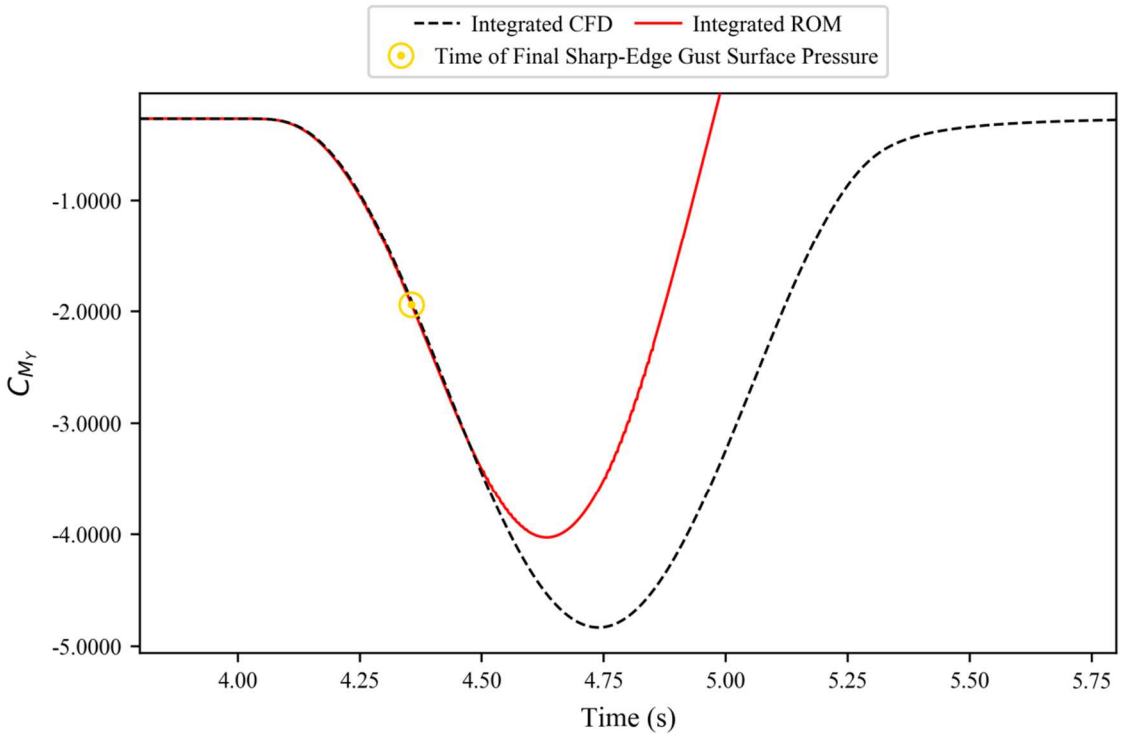

Figure 7.63. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.356 seconds.

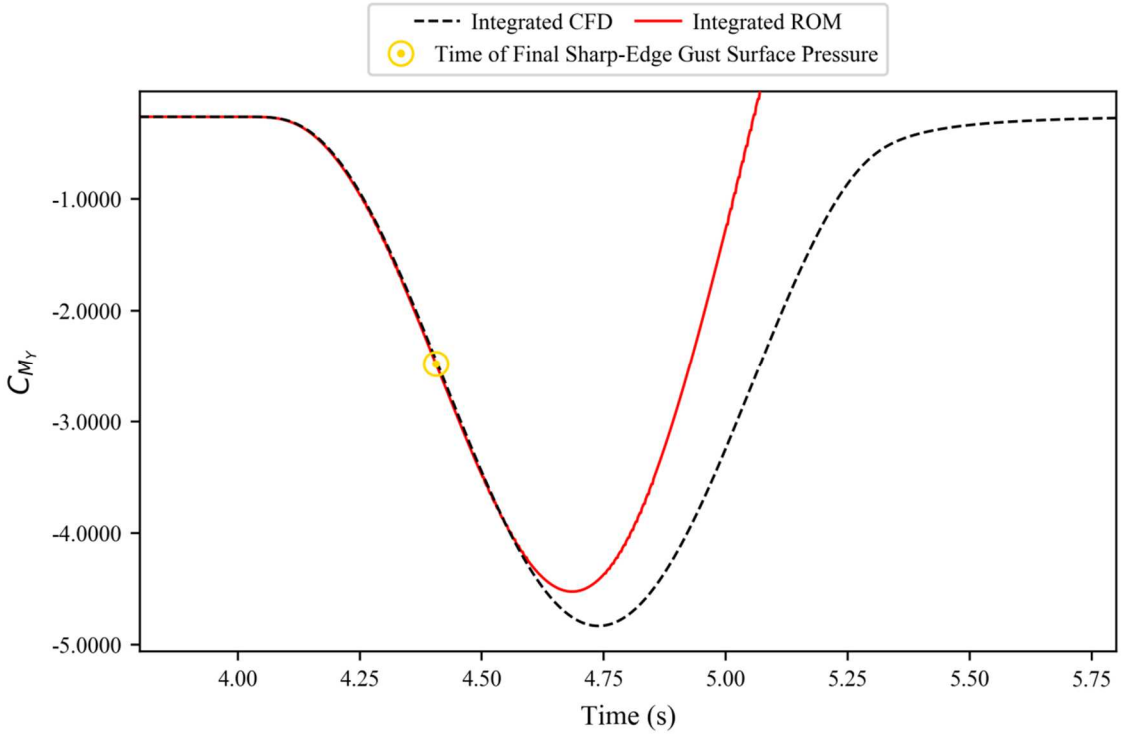

Figure 7.64. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.406 seconds.

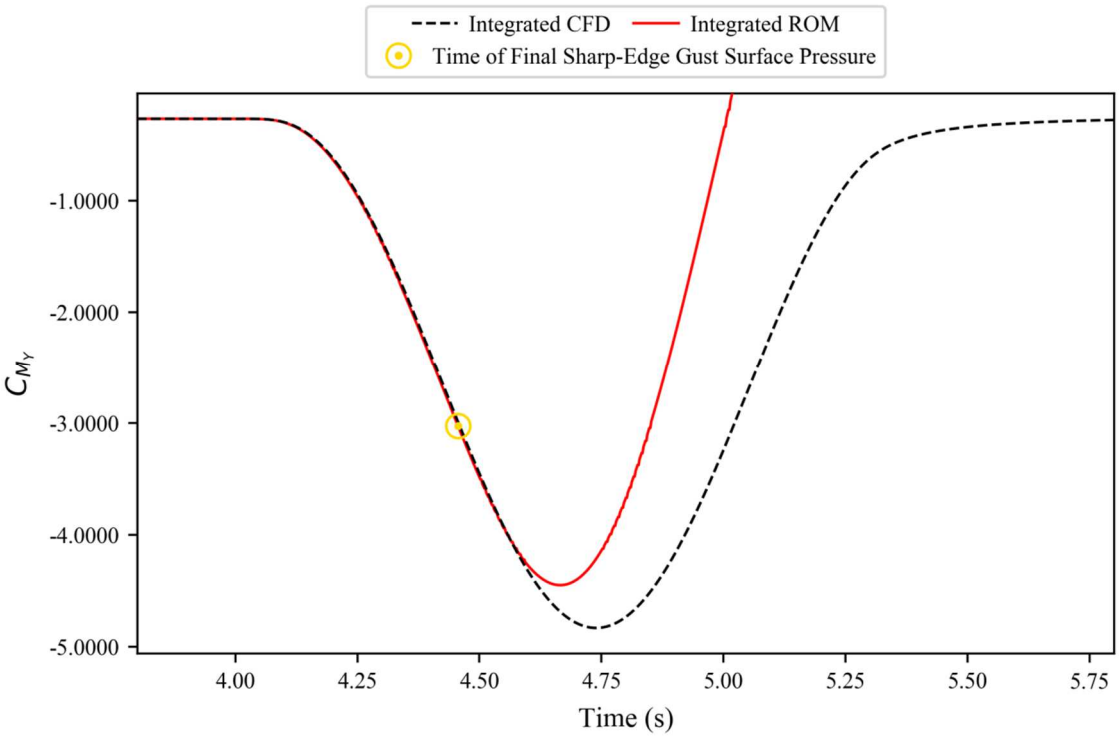

Figure 7.65. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.456 seconds.

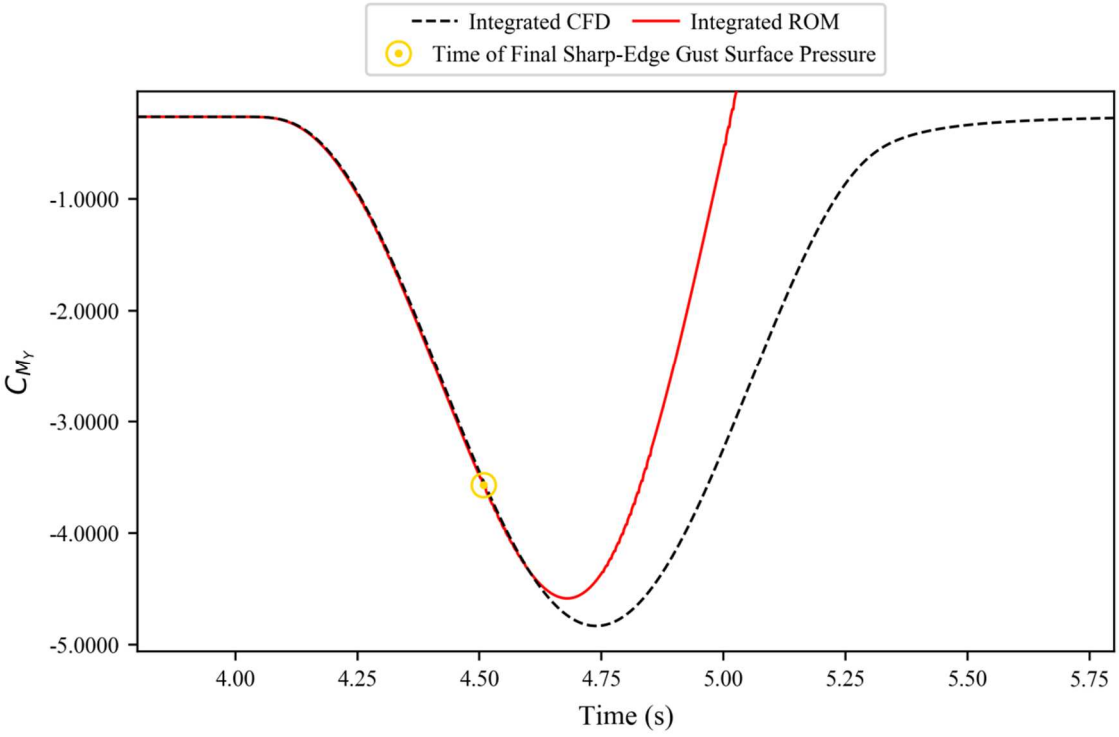

Figure 7.66. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 1, gust case 4. Created with a final, sharp-edged gust, surface pressure solution at 4.508 seconds.

Despite the aforementioned problem however, it is important to note that the general method can be seen to work; even if far more sharp-edged time steps are required than

perhaps desired. Additionally, the Flight Point 1 case is almost a worst case scenario due to its relatively low Mach number of 0.5. When the Mach number is higher, this problem becomes less of an issue, with the critical time occurring further downstream from the output gusts. This can be seen in the results from Flight Point 3 (Figures 7.67-7.74) which show that, whilst the critical time problem is certainly not desirable, it doesn't prevent the method from working and producing results with a high level of accuracy. So whilst the method would benefit from further development and improvement, it is important to note that, fundamentally, the method still works and would still provide large computational savings when compared to full order CFD.

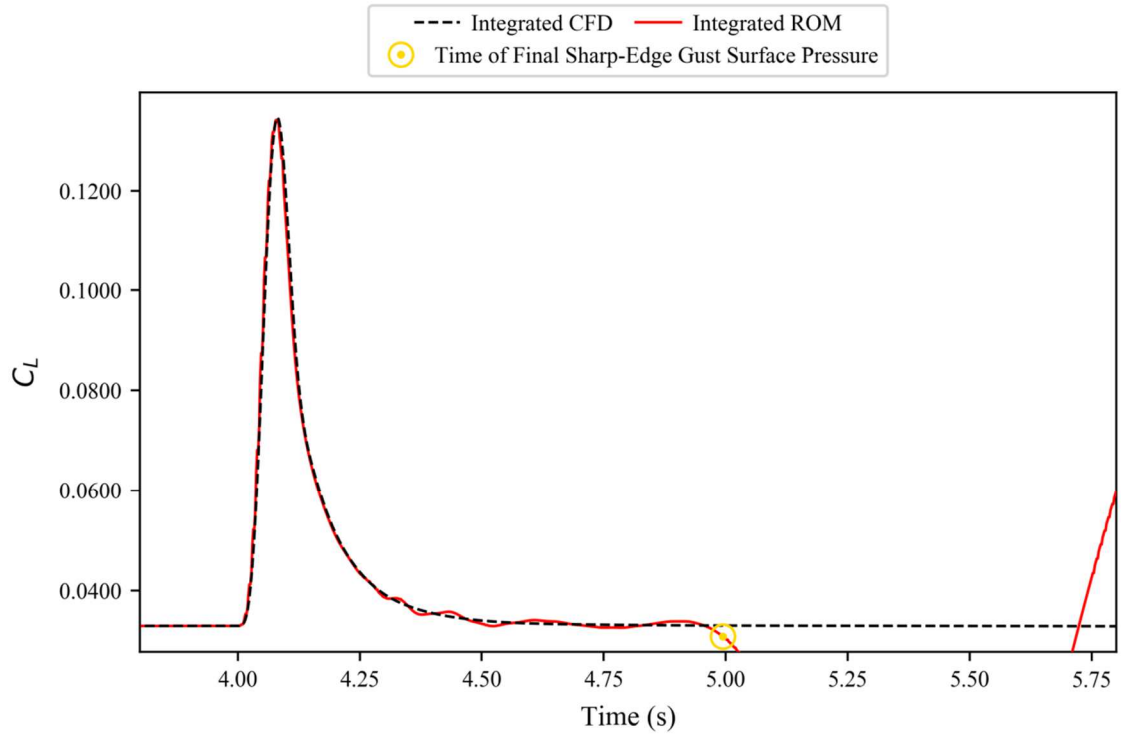

Figure 7.67. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 1.

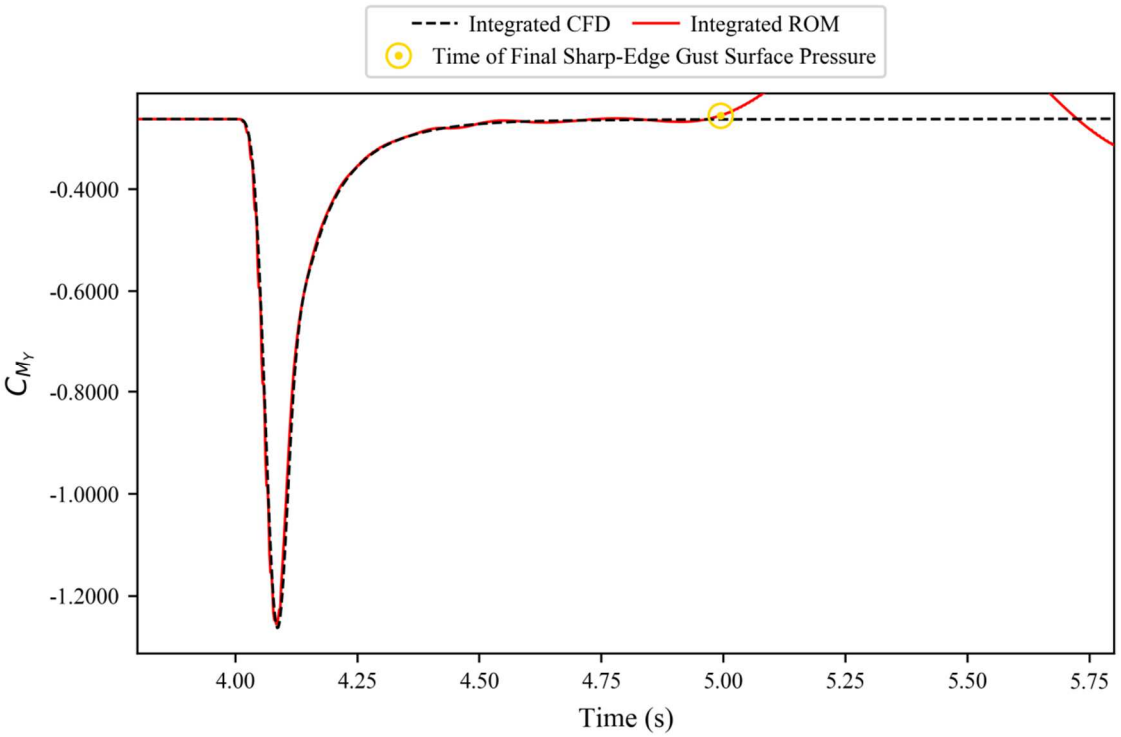

Figure 7.68. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 1.

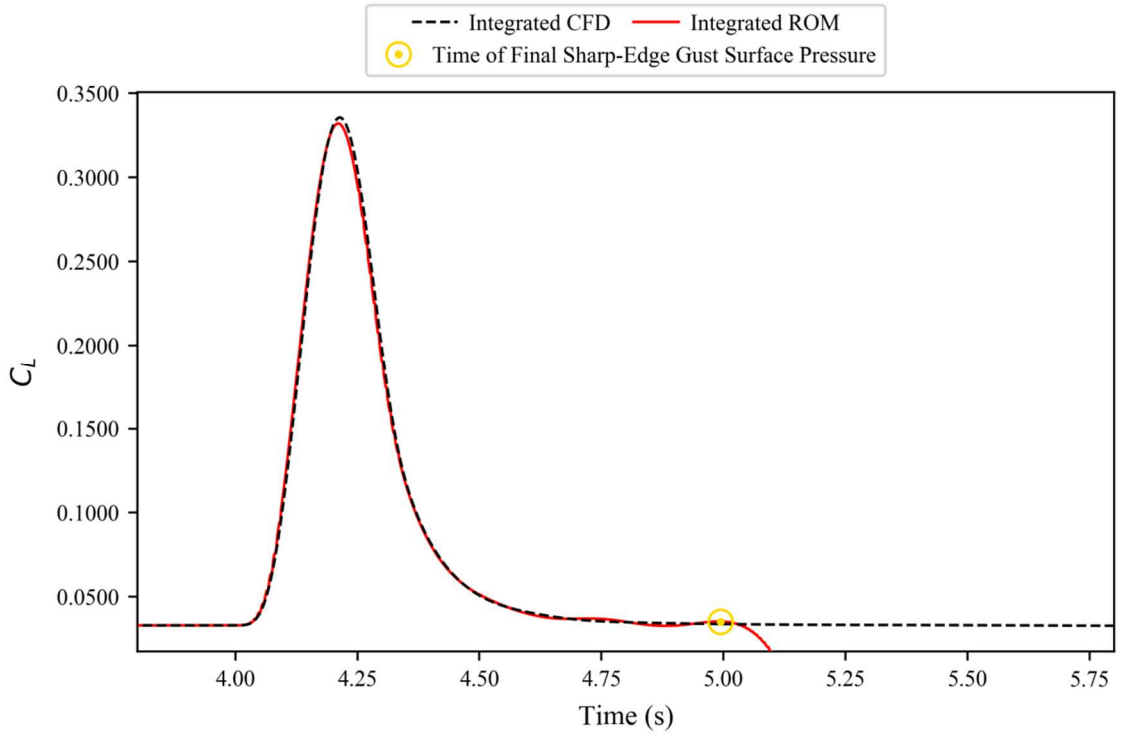

Figure 7.69. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 2.

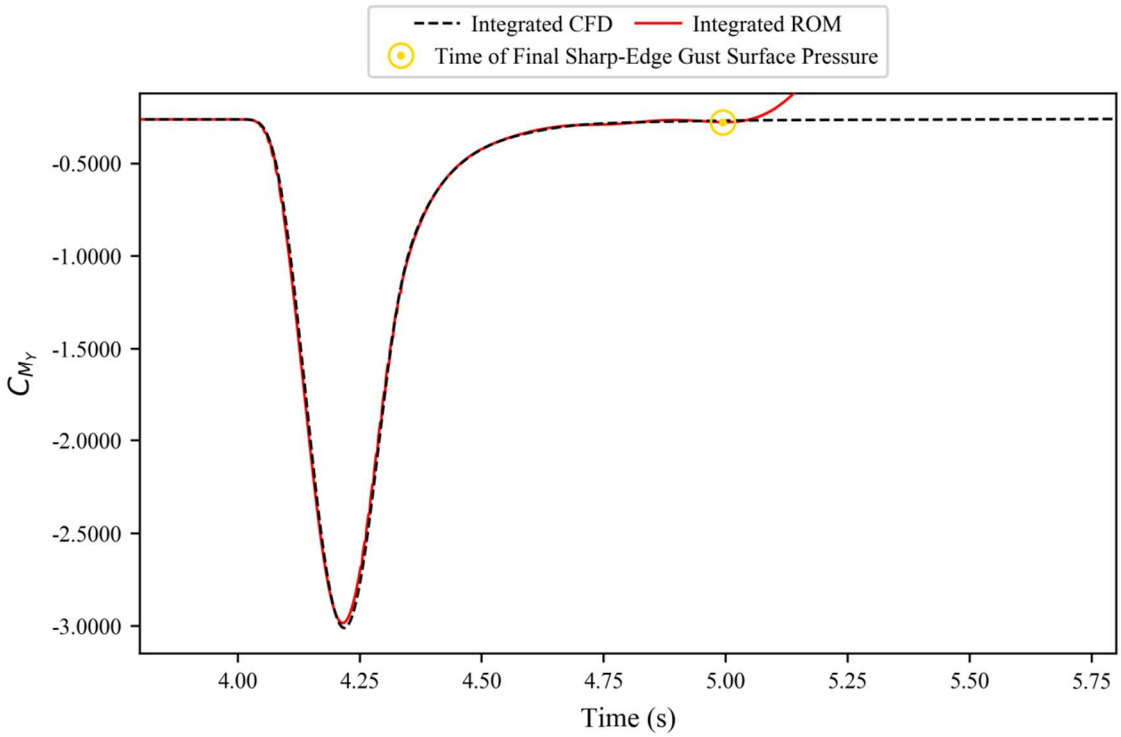

Figure 7.70. Coefficient of pitching, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 2.

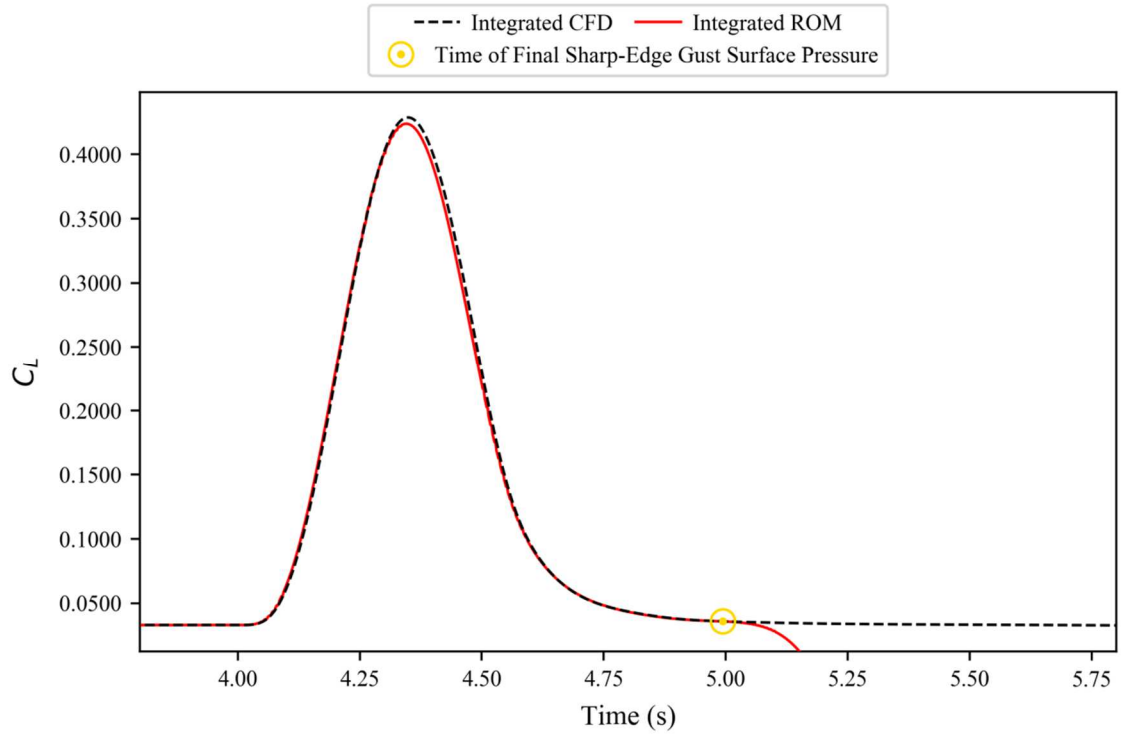

Figure 7.71. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 3.

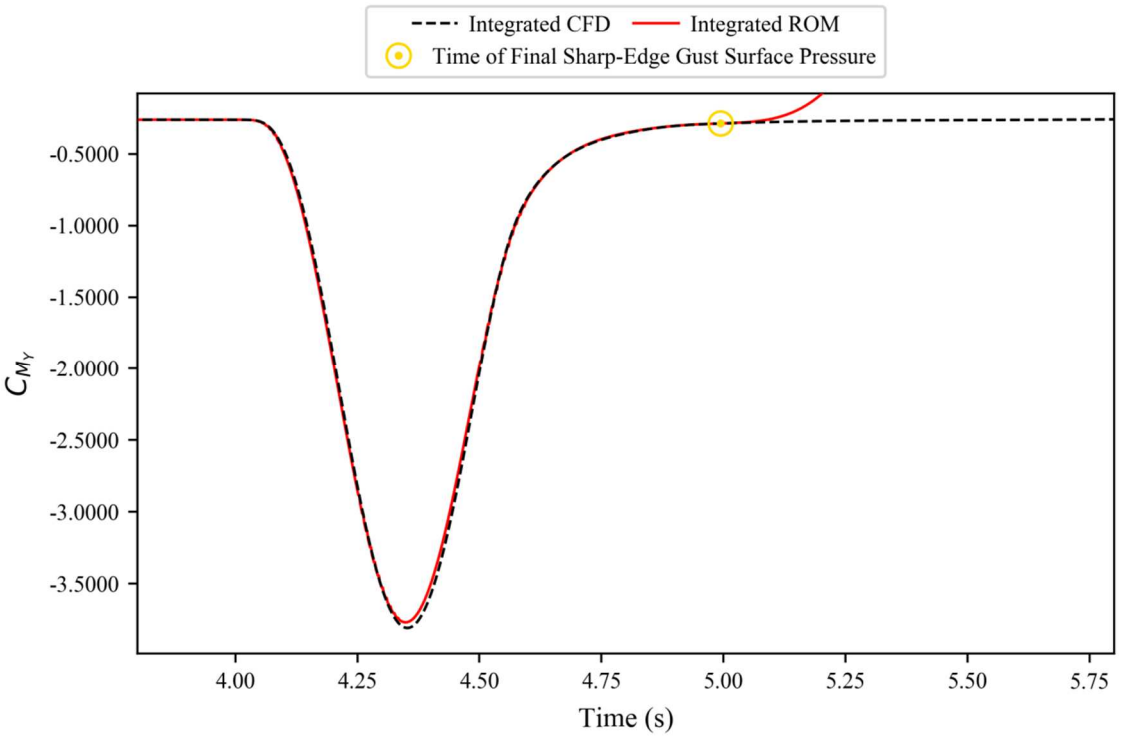

Figure 7.72. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 3.

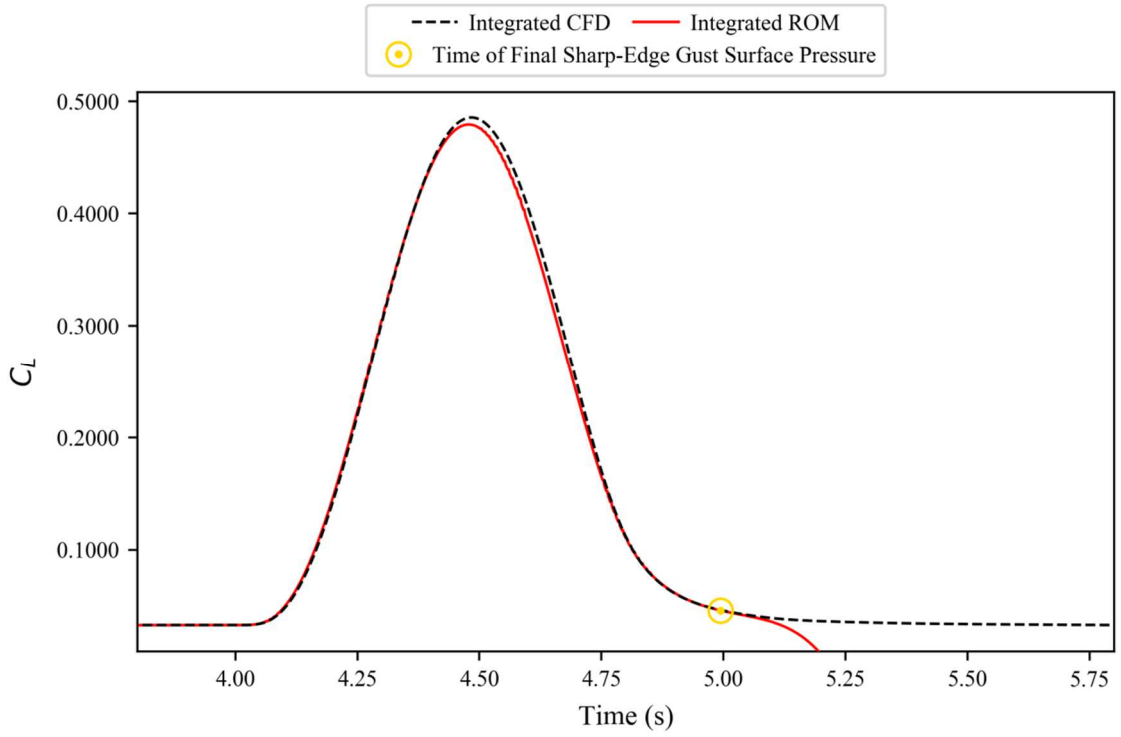

Figure 7.73. Coefficient of lift, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 4.

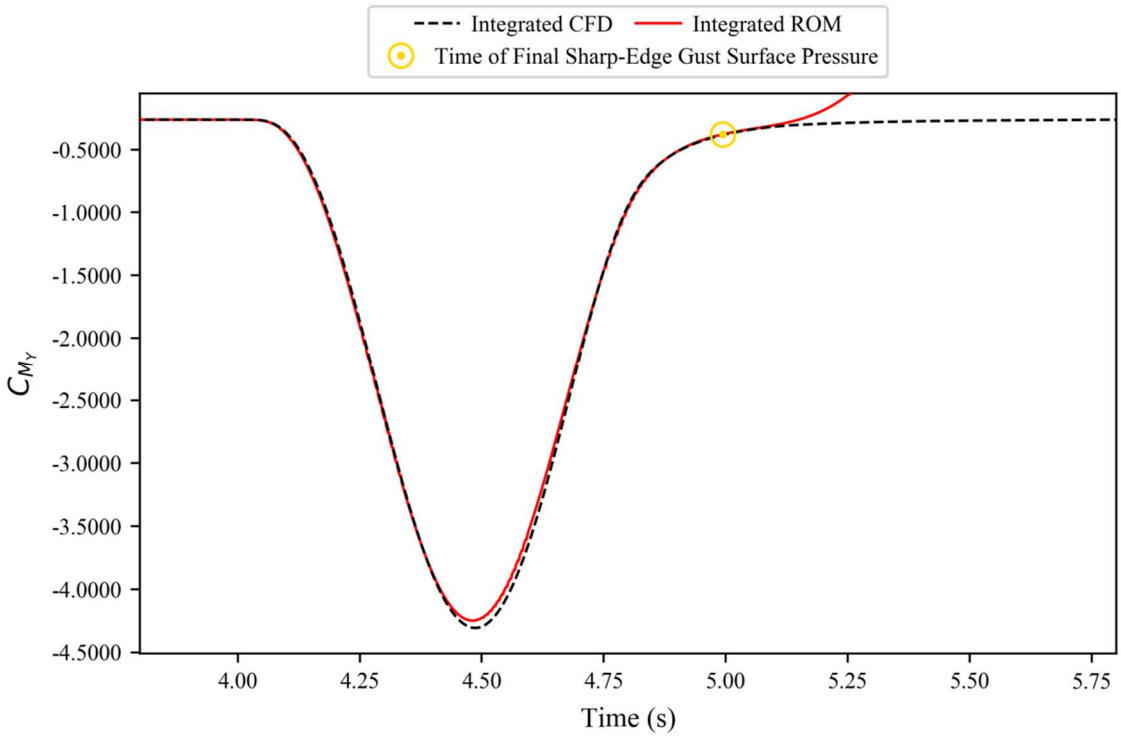

Figure 7.74. Coefficient of pitching moment, calculated by integrating surface pressures in strips, for the FFAST wing at flight point 3, gust case 4.

7.3. Use within Gust Reconstruction Process

The aims of the work carried out within this thesis (see Section 1.6.2) focused on the use of the ROM within the early design stages of aircraft. Therefore the ROM methods put forward have, where beneficial, been tailored to this application. However, it is important to note that this does not prevent any of the ROM methods put forward being used in other applications.

A notable example of the ROM methods being applied to a process other than the early design of aircraft, is the work carried out by Simeone *et al* [148]. The authors put forward a method by which the shape of a gust could be reconstructed based on the system response to it. The long term goals of this work was to develop a method which could be used in real (or near real) time on an aircraft in flight. Given that gusts are often critical design cases, it was hoped by the authors that this method would be able to assist maintenance crews on the ground by making them aware of any gusts encountered that could have caused damage to the aircraft structure. This information could then be coupled with existing aeroelastic CFD simulations to also identify where

any possible damage was likely to have occurred. It was hoped that as a result, the workload on maintenance crews could be substantially reduced.

The work carried out within the paper focused on testing the reconstruction method put forward on a two-dimensional aerofoil. To see the effect different gust modelling methods had, the authors used the Unsteady Lumped Vortex Method (ULVM) on a simplified flat plate in potential flow, SVM based full order CFD and the ROM put forward in Chapter 0 of this thesis.

To calculate the gust input, first an initial guess is made using a set of weighted parametric functions; this produces a gust response which is compared to the gust being reconstructed. Next, an optimisation algorithm is used to modify the weights of the parametric function. This process is then repeated until the gust output matches the gust being reconstructed. In order to fully test the model, two types of parametric functions were used; Radial Basis Functions (RBF) and Hicks-Henne Bump Functions (HHBF).

The ULVM took approximately 20 minutes to converge on a solution using HHBF; however the accuracy at the start and end of the gust (when the gust velocity was zero) was poor. When using RBF the time to converge increased to approximately 2 hours, however this did produce a near perfect result. Conversely, the SVM based full order CFD method produced a near perfect result when using HHBF, but when using RBF suffered from degraded accuracy at the start and end of the gust (when the gust velocity was zero) as well as the peak of the response; and in both instances took approximately 40 hours to converge. Finally, the ROM (Figures 7.75-7.78) was noted to produce perfect results using either HHBF or RBF, and in both cases converged on a solution significantly quicker than either the ULVM or CFD methods; requiring approximately 55 seconds using RBF and approximately 40 seconds when using HHBF. It should be noted that the ROM does involve an upfront computation cost to build, however given that the long term goal of the project is to apply this method in real (or near real) time on aircraft in flight, this upfront cost can be neglected when making comparisons between the different gust modelling techniques. Thus, for the simple two-dimensional test case, the ROM converged (to a near prefect reconstruction) approximately 180 times quicker than the next fastest method, and over 3,500 times quicker than full order CFD. Furthermore, it should be noted that for more complex cases (such as three-dimensional, aeroelastic, etc.) the ROM convergence time will not notably change,

whereas the other methods would likely see a dramatic increase; and the second fastest method, ULVM, would not be suitable due to how it represents the model as a flat plate.

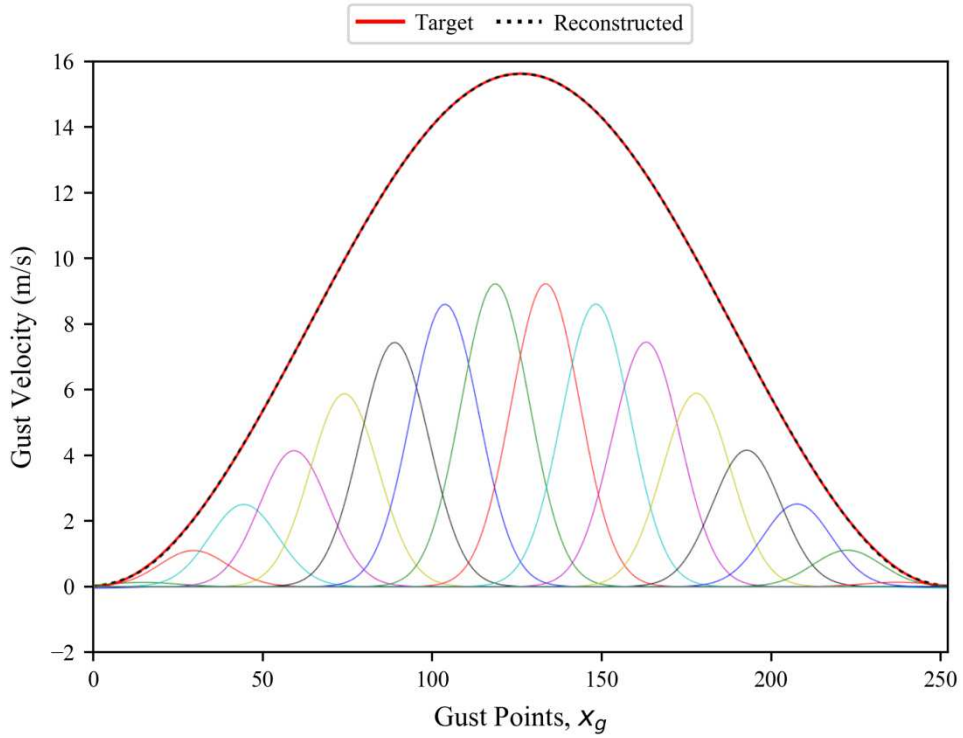

Figure 7.75. ROM based reconstructed gust input using 14 Radial Basis Functions.

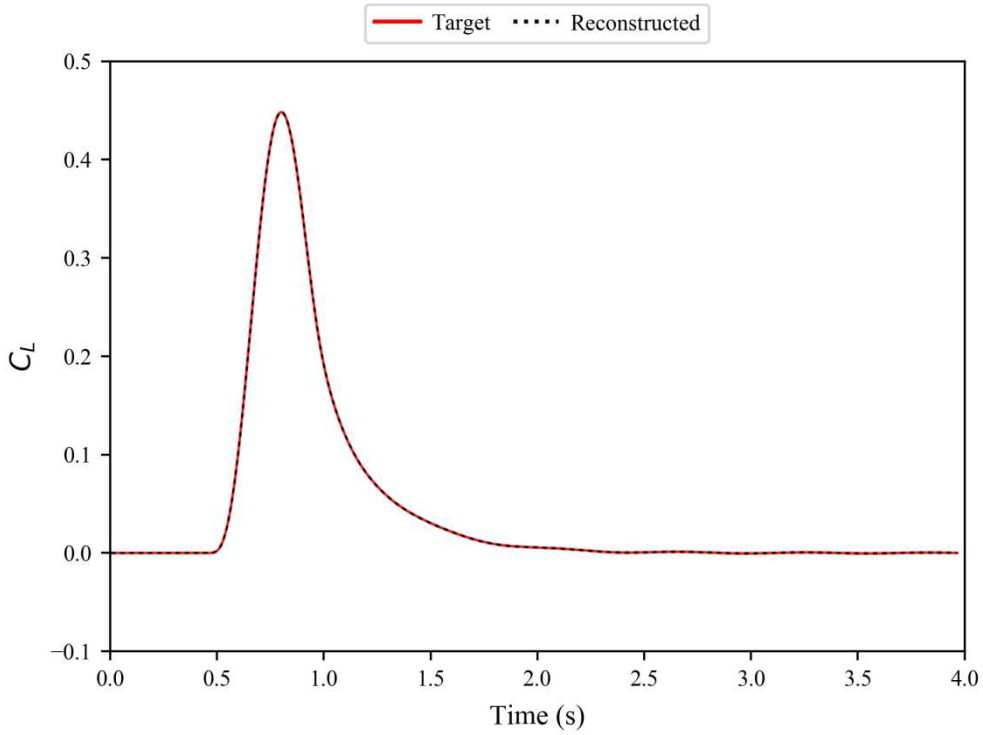

Figure 7.76. ROM based reconstructed gust response using 14 Radial Basis Functions.

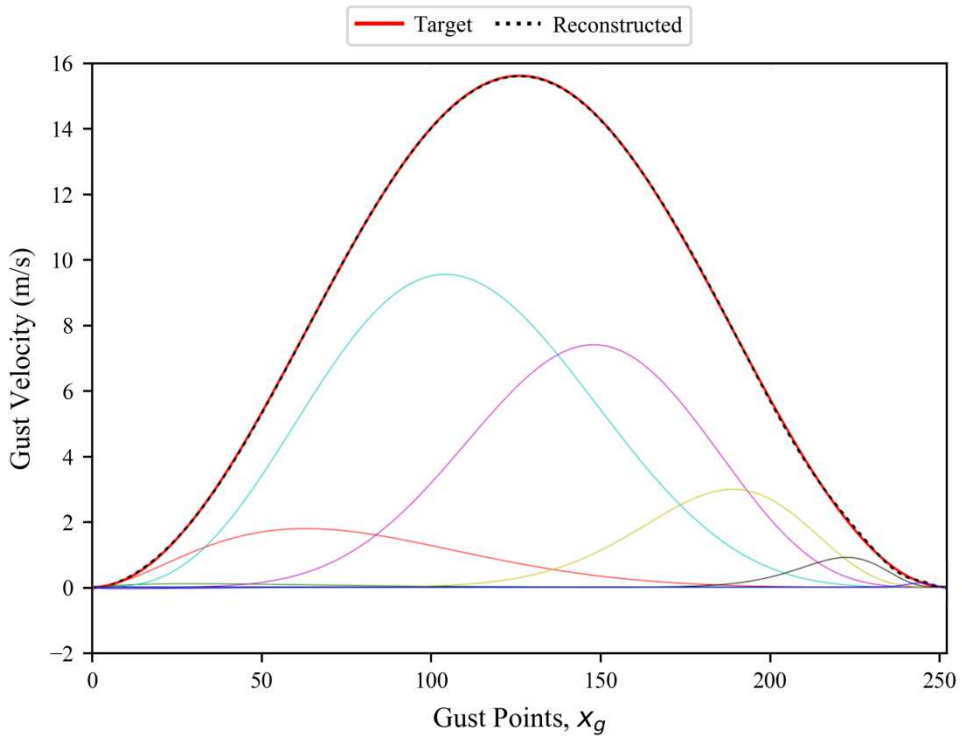

Figure 7.77. ROM based reconstructed gust input using 14 Hicks-Henne Bump Functions.

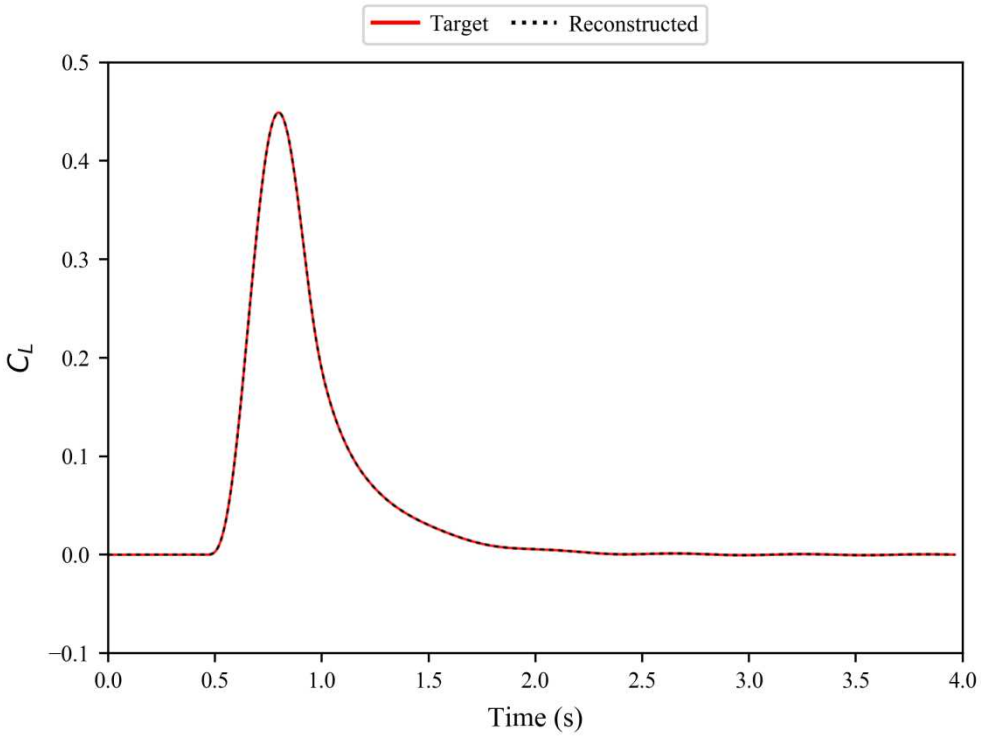

Figure 7.78. ROM based reconstructed gust response using 8 Hicks-Henne Bump Functions.

Therefore, it is clear that the ROM methods put forward in this thesis are not only suitable for their intended purpose, but are also extremely useful in other applications; and are already being used in research being conducted by others.

8. Thesis Conclusions

8.1. Conclusions

In this thesis a series of ROM methods have been developed to model the system response of an object during a gust encounter. The ROM methods have been designed, tested and improved. The main criteria by which the ROM methods were judged were their accuracy and computational cost; in both cases relative to full order CFD simulations and/or the first baseline ROM method. This has led to a final ROM method that would be suitable for industrial applications.

The computational cost of constructing the ROMs was greatly improved (whilst maintaining accuracy), relative to the baseline ROM method, by reducing the number of sharp-edged gusts required from two to one. Further improvements were then made by minimising the number of time steps used within the single sharp-edged gust based ROM method. When propagating the sharp-edged gust from outside the domain to the start of the model, it was shown that this can be achieved by initially using large time steps before transitioning to the normal, smaller, ones. It was also shown that the transition point generally has little effect on the accuracy of the results, so long as it occurs before the start of the model.

Unlike the large-to-small time step transition point, the effect of reducing the number of time steps post-impact was shown to have a large influence on the accuracy of the ROM outputs. This sensitivity to the post-impact length of the sharp-edged gust was mitigated slightly by first calculating the linear gradients from the steady solutions rather than the sharp-edged gust results; and then by modifying the ROM method to be built using an effective stepdown response rather than a pulse response (which was obtained from the sharp-edged gust solution). This new method of constructing a ROM, using a stepdown response rather than a pulse input, significantly improved the performance of the ROMs. Whilst maintaining a suitably high level of accuracy, the stepdown based ROMs required dramatically less data; with the sharp edge gust going from having to be run for just 2-2.33 model lengths (post impact); down from 7-9 model lengths for the best pulse response based ROM.

The most notable limitation is the breakdown in the accuracy of the ROM that occurred when the system response became highly non-linear. In this instance the ROM method was shown to work well until used in extreme test cases; which are likely beyond what might reasonably be expected to be encountered in an industrial application. As such, the reduction of accuracy in these extreme cases should be noted, but the ROM is still highly useful. It is important to view this limitation in the context of its primary intended application; early aircraft design. Due to the high computational cost of full order CFD, it is currently common to use simpler methods to get an overview of which gust cases may define the loads envelope. Whilst these methods are computational inexpensive (relative to CFD) their accuracy is also substantially decreased; and as such, even the worst results obtained from the ROMs in this thesis are likely to either match or exceed existing levels of accuracy.

ROM stability was initially achieved solely through the application of the restarting method. However, it was shown that whilst this method was reasonably robust, there were cases were it worsened the results considerably. As a result, an alternative method known as Schur mirroring was explored. This was shown to be more robust than restarting; however it was also noted that it reduced the accuracy of the ROM slightly when compared to restarting. As such, the two methods were used in tandem, with restarting acting as the primary stability method, and Schur mirroring being used in cases where restarting failed; thus maintaining the highest accuracy levels possible, whilst also guaranteeing a viable ROM is produced. The stability of the surface pressure reconstruction is slightly more complex, and due to time limitations has not been resolved at the time of writing. However, this looks to be closely tied to the number of sharp-edged gust time steps for which the surface pressures are available; and as such this limitation could be mitigated in the short term by sacrificing some of the computational savings and running the sharp-edged gust for longer. Ultimately, the true impact of this particular limitation is unknown, however it is highly likely that a stability method (much like restarting or Schur mirroring) could be developed or adapted to resolve this problem.

The final ROM method shows extremely high potential for use in early aircraft design application, and for other applications such as gust reconstruction. It is impossible to fully quantify computational savings due this is relative to the number of gusts being modelled, each gust requiring different numbers of time steps, varying iterations needed

per time step, etc. However, if the computational cost is considered to be directly proportional to the number of unsteady time steps required, then the final ROM method has been shown to offer a 94.78% reduction in computational cost compared to the baseline ROM method. Then, as the baseline ROM method was roughly equivalent to the computational cost needed to model a single '1-cosine' gust using full order CFD, then the final ROM method offers computational savings (relative to full order CFD) of approximately 98.7% if modelling 4 gusts at a single flight point. Final computational savings were made possible via the altitude adaptations made, which yielded a reduction in computational costs (relative to full order CFD) of approximately 99.4% when modelling 4 gusts at each of 3 different altitudes (with a constant Mach number). Coupled with a level of accuracy that is typically very close to full order CFD and these extremely large computational savings make the final ROM method(s) ideally suited for the original application. This in turn couples with the other key strength of the ROM method, being able to model additional gusts at negligible computational cost, to produce a versatile piece of software that can be exploited in other areas; as demonstrated by its application in gust reconstruction.

Finally, the ROM method was modified so that the surface pressures for the output gust could be produced in a post-process where the only additional data required was the surface pressures of the sharp-edged gust. This was shown to suffer slightly from a stability issue however the results were mostly a very strong match to those obtained via full order CFD. Therefore, whilst the current process still requires further work, the reconstruction of the surface pressures for the output gust demonstrate the potential for the ROM method to be integrated into existing industrial processes; which often directly use surface pressure results. The results currently require more data than is ideally desired, but the ROM was still capable of producing near CFD levels of accuracy for a fraction of the computational cost. As such, this demonstrates that with a small amount of further development, the ROM method could be integrated into existing processes to allow for them to be used earlier in the design process than is currently feasible.

8.2. Future Work

Perhaps the most likely future development of the ROM method will focus on the improvement of the surface pressure reconstruction method. Whilst the method was

shown to work and produce highly accurate results, currently the method shows a new type of stability issue. This was shown to be worked around by increasing the number of sharp-edged gust time steps; however this is far from ideal. As such, fully investigating and resolving this issue would be a logical progression.

Another area that could be further explored is the effect of static deflections in the altitude adjustment method. The method put forward in this thesis was only tested on the rigid wing model, and whilst it highlighted that the method works for such a case, the lack of aeroelastic deformation could mask a potential issue. Static deflections scale directly with free stream pressure, and as such as the altitude changes so too will the trim shape of the model. As highlighted by Heinrich *et al* [47], the effect of trim shape can be significant in the results obtained, and as such this could be a notable weakness in the method put forward. However, without testing the method directly, it is not possible to know the exact impact on the accuracy of the method. Therefore, carrying out such testing could allow the method to be further validated or else potentially improved and made more robust.

The ROM method was designed, tested and improved with the application of use within the early design stages of aircraft, in mind. As such, there likely exists ample opportunity to further adapt, develop and implement the ROM method in other applications. This was demonstrated to good effect through the application of the ROM method within a gust reconstruction process. In this process the ROM method was shown to massively outperform other methods without any tailoring of it to this purpose. As this process used the ROM method in a way it wasn't explicitly designed for (rapid calculation of multiple gust responses), it is highly likely that its performance in this type of application could be improved. Likewise, with some modifications it is likely that the ROM method could be adapted to perform equally well in other applications besides.

Finally, for the flight mechanics test case, the gusts used were effectively worst case scenarios; with no gust alleviation used. As such, the degradation in accuracy is not particularly worrisome. However, these results do still demonstrate that the ROM method, built around linearising near-linear responses, does start to fall down as the response becomes highly non-linear. As such, exploring the full extent of this issue, and

developing methods by which it could be alleviated, would be a logical progression to the work carried out within this thesis.

9. References

1. Wright J. R., Cooper J. E. *Introduction to Aircraft Aeroelasticity and Loads*. Second Edi. Chichester, West Sussex: Wiley; 2015.

2. European Aviation Safety Agency. *Certification Specificaitons and Acceptable Means of Compliance for Large Aeroplanes - CS-25* [Internet]. 2016 [cited 2017 May 7]. p. 1–885. Available from: https://www.easa.europa.eu/system/files/dfu/CS-25 Amendment 18_0.pdf

3. Wales C., Gaitonde A. L., Jones D. P. *Reduced Order Modelling for Aeroelastic Aerofoil Response to a Gust*. In: 51st AIAA Aerospace Sciences Meeting. Grapevine (TX), USA: American Institute of Aeronautics and Astronautics; 2013. p. 1–16.

4. Raveh D. E., Zaide A. *Numerical Simulation and Reduced-Order Modeling of Airfoil Gust Response*. AIAA J. 2006;44(8):1826–34.

5. Wales C., Gaitonde A. L., Jones D. P. *Reduced Order Modelling for the Gust Response of the Ffast Wing*. In: IFASD 2013 - International Forum on Aeroelasticity and Structural Dynamics. Bristol, UK; 2013.

6. Wales C., Gaitonde A. L., Jones D. P. *Stabilisation of Reduced Order Models via Restarting*. In: 15th International Forum on Aeroelasticity and Structural Dynamics. Paris, France; 2011. p. 578–99.

7. Mahapatra P. R. *Aviation Weather Surveillance System*. Exeter, UK: The Institution of Electrical Engineers; 1999.

8. Peery D. J., Azar J. J. *Aircraft Structures*. Second Edi. New York (NY), USA: McGraw-Hill; 1982.

9. Yates (Jr) E. C. *Modified-Strip-Analysis Method for Predicting Wing Flutter at Subsonic to Hypersonic Speeds*. J Aircr. 1966;3(1):25–9.

10. Wagner H. A. *Origin of the Dynamic Lift of Wings*. Technical University of Berlin; 1924.

11. Küssner H. G. *Zusammenfassender Bericht Über Den Instationären Auftrieb von Flügeln*. Luftfahrtforschung. 1936;13(12):410–24.

12. Clancy L. J. *Aerodynamics*. London, UK: Pitman Books; 1975.

13. Albano E., Rodden W. P. *A Doublet-Lattice Method for Calculating Lift Distributions on Oscillating Surfaces in Subsonic Flows*. AIAA J. 1969;7(2):279–85.

14. Blair M. *A Compilation of the Mathematics Leading to the Doublet Lattice Method*. 1992.

15. Baals D. D., Corliss W. R. *Wind Tunnels of NASA*. Washington D.C., USA; 1981.

16. Tang D. M., Cizmas P. G. A., Dowell E. H. *Experiments and Analysis for a Gust Generator in a Wind Tunnel*. J Aircr. 1996;33(1):139–48.

17. Mai H., Neumann J., Hennings H. *Gust Response: A Validation Experiment and Preliminary Numcerical Simulations*. In: 15th International Forum on Aeroelasticity and Structural Dynamics (IFASD). Paris, France; 2011. p. 1–20.

18. Allen N. J., Quinn M. K. *Development of a Transonic Gust Rig for Simulation of Vertical Gusts on Half-Models*. In: 31st AIAA Aerodynamic Measurement Technology and Ground Testing Conference. Dallas (TX), USA; 2015. p. 1–12.

19. Institution of Engineering Designers. *Gust of Change*. Engineering Designer [Internet]. 2017;20–1. Available from: http://www.ara.co.uk/assets/Uploads/42T-20170113-Engineering-Designer-Gust-rig-feature-with-front-cover.pdf

20. Aircraft Research Association. *About ARA* [Internet]. [cited 2017 Aug 25]. Available from: http://www.ara.co.uk/about-ara/

21. Richardson L. F. *On the Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam*. Philos Trans R Soc London Ser A, Contain Pap a Math or Phys Character. 1911;210:307–57.

22. Turner M. J., Clough R. W., Martin H. C., Topp L. J. *Stiffness and Deflection Analysis of Complex Structures*. J Aeronaut Sci. 1956;23(9):805–23.

23. Johnson N. L. *The Legacy and Future of CFD At Los Alamos*. In: 1996 Canadian CFD Conference. Ottawa, Canada; 1996. p. 1–20.

24. Witherden F. D. *Future Directions of Computational Fluid Dynamics*. In: 23rd AIAA Computational Fluid Dynamics Conference. Denver (CO), USA; 2017. p. 1–16.

25. Moretti G., Abbett M. *A Time-Dependent Computational Method for Blunt Body Flows*. AIAA J. 1996;4(12):2136–41.

26. Maccormack R. W. *The Effect of Viscosity in Hypervelocity Impact Cratering*. In: 4th Aerodynamic Testing Conference. Cincinnati (Oh), USA; 1969.

27. Lax P., Wendroff B. *Systems of Conservation Laws*. Commun Pure Appl Math. 1960;13(2):217–37.

28. Murman E. M., Cole J. D. *Calculation of Plane Steady Transonic Flows*. AIAA J. 1971;9(1):114–21.

29. Caughey D. A., Jameson A. *Development of Computational Techniques for Transonic Flows: An Historical Perspective*. In: IUTAM Symposium Transsonicum IV. Göttingen, Germany; 2002. p. 183–94.

30. Jameson A. *Iterative Solution of Transonic Flows over Airfoils and Wings, Including Flows at Mach 1*. Commun Pure Appl Math. 1974;27(3):283–309.

31. Bakhvalov N. S. *On the Convergence of a Relaxation Method with Natural*

Constraints on an Elliptic Operator. USSR Comput Math Math Phys. 1966;6(5):101–35.

32. Fedorenko R. P. *A Relaxation Method for Solving Elliptic Difference Equations*. USSR Comput Math Math Phys. 1962;1(4):1092–6.

33. Fedorenko R. P. *The Speed of Convergence of an Iteration Process*. USSR Comput Math Math Phys. 1964;4(3):227–35.

34. Brandt A. *Multi-Level Adaptive Solutions to Boundary-Value Problems*. Math Comput. 1977;31(138):333–90.

35. Yavneh I. *Why Multigrid Methods Are so Efficient*. Comput Sci Eng. 2006;8:12–22.

36. Jameson A., Baker T. J., Weatherill N. P. *Calculation of Inviscid Transonic Flow Over a Complete Aircraft*. In: 24th Aerospace Sciences Meeting. Reno (NV), USA; 1986. p. 1–13.

37. Simon H. D. *Partitioning of Unstructured Problems for Parallel Processing*. Comput Syst Eng. 1991;2(2–3):135–48.

38. Tang L., Baeder J. D. *Adaptive Euler Simulations of Airfoil-Vortex Interaction*. Int J Numer Methods Fluids. 2007;53(5):777–92.

39. Steger J. L., Dougherty F. C., Benek J. A. *A Chimera Grid Scheme*. Am Soc Mech Eng Fluid Eng Div. 1983;5:55–69.

40. Hixon R., Golubev V., Mankbadi R. R., Scott J. R., Sawyer S., Nallasamy M. *Application of Nonlinear Computational Aeroacoustics Code to the Gust-Airfoil Problem.* AIAA J. 2006;44(2):323–8.

41. Parameswaran V., Baeder J. D. *Indicial Aerodynamics in Compressible Flow - Direct Computational Fluid Dynamics Calculations*. J Aircr. 1997;34(1):131–3.

42. Singh R., Baeder J. D. *Direct Calculation of Three Dimensional Indicial Lift Responses Using Computational Fluid Dynamics*. J Aircr. 1997;34(4):465–71.

43. Singh R., Baeder J. D. *Generalized Moving Gust Response Using CFD with Application to Airfoil-Vortex Interaction*. In: AIAA 15th Applied Aerodynamics Conference. Atlanta (GA), USA; 1997. p. 1–11.

44. Wales C., Jones D., Gaitonde A. *Prescribed Velocity Method for Simulation of Aerofoil Gust Responses*. J Aircr. 2015;52(1):64–76.

45. National Aeronautics and Space Administration. *Overview of CFD Verification and Validation* [Internet]. 2008 [cited 2017 Aug 28]. Available from: https://www.grc.nasa.gov/www/wind/valid/tutorial/overview.html

46. National Aeronautics and Space Administration. *Validation Assessment* [Internet]. 2008 [cited 2017 Aug 28]. Available from: https://www.grc.nasa.gov/www/wind/valid/tutorial/valassess.html

47. Heinrich R., Kroll N. *Fluid-Structure Coupling for Aerodynamic Analysis and Design: A DLR Perspective*. In: 46th AIAA Aerospace Sciences Meeting. Reno (NV), USA; 2008. p. 1–31.

48. Moore B. C. *Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction*. IEEE Trans Automat Contr. 1981;26(1):17–32.

49. Karpel M. *Design for Active Flutter Suppression and Gust Alleviation Using State-Space Aeroelastic Modeling*. J Aircr. 1982;19(3):221–7.

50. Karpel M. *Reduced-Order Aeroelastic Models via Dynamic Residualization*. J Aircr. 1990;27(5):499–455.

51. Karpel M. *Multidisciplinary Optimization of Aeroservoelastic Systems Using Reduced-Size Models*. J Aircr. 1992;29(5):939–46.

52. Karpel M. *Reduced-Order Models for Integrated Aeroservoelastic Optimization*. J Aircr. 1999;36(1):146–55.

53. Hall K. C. *Eigenanalysis of Unsteady Flows about Airfoils, Cascades and Wings*. AIAA J. 1994;32(12):2426–32.

54. Hall K. C., Florea R., Lanzkron P. J. *A Reduced Order Model of Unsteady Flows in Turbomachinery*. J Turbomach. 1995;117(3):375–83.

55. Gallivan K. A., Grimme E. J., Van Dooren P. *Padé Approximation of Large-Scale Dynamic Systems with Lanczos Methods*. In: 33rd Conference on Decision and Control. Lake Buena Vista (FL), USA; 1994. p. 443–8.

56. Dowell E. H. *Eigenmode Analysis in Unsteady Aerodynamics - Reduced-Order Models*. AIAA J. 1996 Aug;34(8):1578–83.

57. Romanowski M. C. *Reduced Order Unsteady Aerodynamic and Aeroelastic Models Using Karhunen-Loeve Eigenmodes*. In: 6th Symposium on Multidisciplinary Analysis and Optimization. Bellevue (WA), USA: American Institute of Aeronautics and Astronautics; 1996. p. 7–13.

58. Romanowski M. C., Dowell E. H. *Reduced Order Euler Equations for Unsteady Aerodynamic Flows - Numerical Techniques*. In: 34th Aerospace Sciences Meeting and Exhibit. Reston (VA), USA: American Institute of Aeronautics and Astronautics; 1996.

59. Wilson E. L., Yuan M.-W., Dickens J. M. *Dynamic Analysis by Direct Superposition of Ritz Vectors*. Earthq Eng Struct Dyn. 1982;10:813–21.

60. Kim T. *Frequency-Domain Karhunen-Loeve Method and Its Application to Linear Dynamic Systems*. AIAA J. 1998;36(11):2117–23.

61. Karhunen K. *Zur Spektraltheorie Stochastischer Prozesse*. Ann Acad Sci Fenn. 1946;37:1–7.

62. Loève M. *Probability Theory*. Princeton (NJ), USA: D. Van Nostrand Company;

1955.

63. Berkooz G., Holmes P., Lumley J. L. *The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows*. Annu Rev Fluid Mech. 1993;25:539–75.

64. Sirovich L. *Turbulence and the Dynamics of Coherent Structures. Part I: Coherent Structures*. Q Appl Math. 1987;45(3):561–71.

65. Lumley J. L. *The Structure of Inhomogeneous Turbulent Flows*. In: Atmospheric Turbulence and Radio Wave Propagation. Moscow, Russia; 1967. p. 166–76.

66. Hall K. C., Thomas J. P., Dowell E. H. *Proper Orthogonal Decomposition Technique for Transonic Unsteady Aerodynamic Flows*. AIAA J. 2000;38(10):1853–62.

67. Willcox K. *Reduced-Order Aerodynamic Models for Aeroelastic Control of Turbomachines*. Massachusetts Institute of Technology; 2000.

68. Willcox K., Peraire J. *Balanced Model Reduction via the Proper Orthogonal Decomposition*. AIAA J. 2002;40(11):2323–30.

69. Schmidt A., Potschka A., Körkel S., Bock H. G. *Derivative-Extended POD Reduced-Order Modelling for Parameter Estimation*. SIAM J Sci Comput. 2013;35(6):2696–717.

70. Rowley C. W. *Model Reduction for Fluids, Using Balanced Proper Orthogonal Decomposition*. Int J Bifurc Chaos. 2005;15(3):997–1013.

71. Lall S., Marsden J. E., Glavaški S. *Empirical Model Reduction of Controlled Nonlinear Systems*. In: 14th International Federation of Automatic Control (IFAC) Congress. Beijing, China; 1999. p. 1–6.

72. Lall S., Marsden J. E., Glavaški S. *A Subspace Approach to Balanced Truncation for Model Reduction of Nonlinear Control Systems*. Int J Robust Nonlinear Control. 2002 May;12:519–35.

73. Bekemeyer P., Ripepi M., Heinrich R., Görtz S. *Nonlinear Unsteady Reduced Order Models Based on Computational Fluid Dynamics for Gust Loads PRedictions*. In: AIAA Applied Aerodynamics Conference. Atlanta (GA), USA; 2018. p. 1–19.

74. Juang J.-N., Pappa R. S. *An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction*. J Guid Control Dyn. 1985;8(5):620–7.

75. Silva W. A., Raveh D. E. *Development of Unsteady Aerodynamic State-Space Models from CFD-Based Pulse Responses*. In: 19th AIAA Applied Aerodynamics Conference. Anaheim (CA), USA: American Institute of Aeronautics and Astronautics; 2001. p. 1–9.

76. Gaitonde A. L., Jones D. P. *Reduced Order State-Space Models from the Pulse Responses of a Linearized CFD Scheme*. Int J Numer Methods Fluids. 2003;42(6):581–606.

77. Wales C., Gaitonde A. L., Jones D. P. *Reduced-Order Modeling of Gust Responses*. J Aircr. 2017;54(4):1350–63.

78. Silva W. A., Bartels R. E. *Development of Reduced-Order Models for Aeroelastic Analysis and Flutter Prediction Using the CFL3Dv6.0 Code*. J Fluids Struct. 2004;19(6):729–45.

79. Kramer B., Gugercin S. *Tangential Interpolation-Based Eigensystem Realization Alogirthm for MIMO Systems*. Math Comput Model Dyn Syst. 2016;22(4):282–306.

80. Ma Z., Ahuja S., Rowley C. W. *Reduced-Order Models for Control of Fluids Using the Eigensystem Realization Algorithm*. Theor Comput Fluid Dyn. 2011;25(1):233–47.

81. Cardoso M. A. *Development and Application of Reduced-Order Modelling Procedures for Reservoir Simulation*. Stanford University; 2009.

82. Kung S.-Y. *A New Identification and Model Reduction Algorithm via Singular Value Decomposition*. In: 12th Asilomar Conference on Circuits, Systems and Computers. Pacific Grove (CA), USA; 1978. p. 705–14.

83. Silva W. A. *Discrete Time Linear and Nonlinear Aerodynamic Impulse Reponses for Efficient CFD Analysis*. College of William & Mary; 1997.

84. Wiener N. *Response of a Nonlinear Device to Noise*. 1942.

85. Silva W. A. *Application of Nonlinear Systems Theory to Transonic Unsteady Aerodynamic Responses*. J Aircr. 1993;30(5):660–8.

86. Silva W. A. *Extension of Nonlinear Systems Theory to General Frequency Unsteady Transonic Aerodynamic Responses*. In: 34th AIAA Structures, Structural Dynamics, and Materials Conference. Hampton (VA), USA; 1993. p. 2490–503.

87. Silva W. A. *Identification of Linear and Nonlinear Aerodynamic Impulse Reponses Using Digital Filter Techniques*. In: 22nd AIAA Atmospheric Flight Mechanics Conference. New Orleans (LA), USA; 1997. p. 584–97.

88. Silva W. A. *Reduced-Order Models Based on Linear and Nonlinear Aerodynamic Impulse Responses*. In: 40th Structures, Structural Dynamics, and Materials Conference and Exhibit. St. Louis (MI), USA; 1999. p. 638–348.

89. Gaitonde A. L., Jones D. P. *Study of Linear Response Identification Techniques and Reduced-Order Model Generation for a 2D CFD Scheme*. Int J Numer Methods Fluids. 2006;52(12):1361–402.

90. Raveh D. E. *Reduced-Order Models for Nonlinear Unsteady Aerodynamics*. AIAA J. 2001;39(8):1417–29.

91. Raveh D. E., Levy Y., Karpel M. *Aircraft Aeroelastic Analysis and Design Using CFD Based Unsteady Loads*. In: 41st Structures, Structural Dynamics, and Materials Conference and Exhibit. Atlanta (GA), USA; 2000. p. 1–11.

92. Wales C. *FFAST Project: Implementation and Validation of 3D Gust ROMS*. 2013.

93. Lindquist D. R., Gilest M. B. *Validity of Linearized Unsteady Euler Equations with Shock Capturing*. AIAA J. 1994;32(1):1994.

94. Amsallem D., Farhat C. *Interpolation Method for Adapting Reduced-Order Modelsand Application to Aeroelasticity*. AIAA J. 2008;46(7):1803–13.

95. Gaitonde A. L., Jones D. P. *Study of Techniques for Obtaining Continuous Models from 2D Discrete Reduced-Order State-Space CFD Models*. Int J Numer Methods Fluids. 2006;52(11):1247–75.

96. Grimme E. J., Sorensen D. C., Van Dooren P. *Model Reduction of State Space Systems via Implicit Restarted Lanczos Method*. Numer Algorithms. 1996;12:1–31.

97. Jaimoukha I. M., Kasenally E. M. *Implicitly Restarted Krylov Subspace Methods for Stable Partial Realizations*. SIAM J Matrix Anal Appl. 1997;18(3):633–52.

98. McKelvey T., Akçay H., Ljung L. *Subspace-Based Multivariable System Identification from Frequency Response Data*. IEEE Trans Automat Contr. 1996;41(7):960–79.

99. Wales C. *FFAST Project: 3D Gust ROM Tool Creating ROM Primary Routines*. 2013.

100. German Aerospace Center (DLR). *DLR TAU Code*. Cologne, Germany; 2015.

101. The MathWorks I. *MATLAB*. Natick (MA), USA; 2015.

102. Anderson J. D. *A History of Aerodynamics*. Cambridge, UK: Cambridge University Press; 1998.

103. Wilson D. C. *Basic Fluid Mechanics*. Second Edi. La Cañada Flintridge (CA), USA: DCW Industries; 2003.

104. BALDWIN B., LOMAX H. *Thin-Layer Approximation and Algebraic Model for Separated Turbulentflows*. In: 16th Aerospace Sciences Meeting. Huntsville (AL), USA: American Institute of Aeronautics and Astronautics; 1978. p. 1–9.

105. Smith A. M. O., Cebeci T. *Numerical Solution of the Turbulent Boundary Layer Equations*. 1967 May.

106. Spalart P. R., Allmaras S. R. *A One-Equation Turbulence Model for Aerodynamic Flows*. In: 30th Aerospace Sciences Meeting and Exhibit. Reno (NV), UAS: American Institute of Aeronautics and Astronautics; 1992. p. 1–23.

107. Spalart P. R., ALLMARAS S. *A One-Equation Turbulence Model for Aerodynamic Flows*. Rech Aerosp. 1994;1:5–21.

108. Spalart P. R. *Trends in Turbulence Treatments*. In: Fluids 2000 Conference and Exhibit. Denver (CO), USA: American Institute of Aeronautics and Astronautics;

2000. p. 1–14.

109. Spalart P. R., Rumsey C. L. *Effective Inflow Conditions for Turbulence Models in Aerodynamic Calculations*. AIAA J. 2007;45(10):2544–53.

110. Allmaras S. R., Johnson F. T., Spalart P. R. *Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model*. In: Seventh International Conference on Computational Fluid Dynamics. Big Island (HI), USA; 2012. p. 1–11.

111. Jones W. P., Launder B. E. *The Prediction of Laminarization with a Two-Equation Model of Turbulence*. Int J Heat Mass Transf. 1972;15(2):301–14.

112. Launder B. E., Sharma B. I. *Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow near a Spinning Disc*. Lett Heat Mass Transf. 1974;1(2):131–7.

113. WILCOX D. C. *Reassessment of the Scale-Determining Equation for Advanced Turbulence Models*. AIAA J. 1988 Nov;26(11):1299–310.

114. Wilcox D. C. *Turbulence Modeling for CFD*. Glendale (CA), USA: DCW Industries; 1993.

115. Menter F. R. *Zonal Two Equation K-w Turbulence Models For Aerodynamic Flows*. In: 23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference. Orlando (FL), USA: American Institute of Aeronautics and Astronautics; 1993. p. 1–22.

116. Menter F. R. *Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications*. AIAA J. 1994;32(8):1598–605.

117. Eça L., Hoekstra M., Hay A., Pelletier D. *A Manufactured Solution for a Two-Dimensional Steady Wall-Bounded Incompressible Turbulent Flow*. Int J Comut Fluid Dyn. 2007;21(3–4):175–88.

118. Aupoix B., Spalart P. R. *Extensions of the Spalart–Allmaras Turbulence Model to Account for Wall Roughness*. Int J Heat Fluid Flow. 2003;24(4):454–62.

119. Rumsey C. L. *Apparent Transition Behavior of Widely-Used Turbulence Models*. Int J Heat Fluid Flow. 2007;28(6):1460–71.

120. Marvriplis D. J. *Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation*. J Comput Phys. 1990;90(2):271–91.

121. Shen W. *An Introduction to Numerical Computation*. Toh Tuck, Singapore: World Scientific Publishing Company Private Limited; 2015.

122. Hrennikoff A. *Solution of Problems of Elasticity by the Framework Method.* J Appl Mech. 1941;8(4):169–75.

123. Courant R. *Variational Methods for the Solution of Problems of Equilibrium and Vibrations*. Bull Am Math Soc. 1943;49(1):1–23.

124. Hirsch C. *Numerical Computation of Internal & External Flows*. Second Edi. Oxford, UK; 2007.

125. Blazek J. *Computational Fluid Dynamics: Principles and Applications*. Oxford, UK; 2001.

126. McDonald P. W. *The Computation of Transonic Flow Through Two-Dimensional Gas Turbine Cascades*. In: ASME 1971 International Gas Turbine Conference and Products Show. Houston (TX), USA: ASME; 1971. p. 1–7.

127. MACCORMACK R., PAULLAY A. *Computational Efficiency Achieved by Time Splitting of Finite Difference Operators*. In: 10th Aerospace Sciences Meeting. San Diego (CA), USA: American Institute of Aeronautics and Astronautics; 1972. p. 1–7.

128. Jameson A. *Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings*. In: AIAA 10th Computational Fluid Dynamics Conference. Honolulu (HI), USA; 1991. p. 1–13.

129. Arnone A., Liou M.-S., Povinelli L. A. *Multigrid Time-Accurate Integration of Navier-Stokes Equations*. In: AIAA 11th Computational Fluid Dynamics Conference. Orlando (FL), USA; 1993. p. 1–9.

130. Gaitonde A. L. *A Dual-Time Method for the Solution of the Unsteady Euler Equations*. Aeronaut J. 1994;98(978):283–91.

131. Collar A. R. *The First Fifty Years of Aeroelasticity*. Aerospace. 1978;February:12–20.

132. Küttler U., Wall W. A. *Fixed-Point Fluid–structure Interaction Solvers with Dynamic Relaxation*. Comput Mech. 2008;43:61–72.

133. Kassiotis C., Ibrahimbegovic A. *Nonlinear Fluid–structure Interaction Problem. Part I: Implicit Partitioned Algorithm, Nonlinear Stability Proof and Validation Examples*. Comput Mech. 2011;47(3):305–23.

134. Beckert A., Wendland H. *Multivariate Interpolation for Fluid-Structure-Interaction Problems Using Radial Basis Functions*. Aerosp Sci Technol. 2001;00:1–11.

135. von Scheven M., Ramm E. *Strong Coupling Schemes for Interaction of Thin-Walled Structures and Incompressible Flows*. Int J Numer Methods Eng. 2011;87:214–31.

136. Matthies H. G., Steindorf J. *Partitioned but Strongly Coupled Iteration Schemes for Nonlinear Fluid–structure Interaction*. Comput Struct. 2002;80:1991–9.

137. Matthies H. G., Steindorf J. *Partitioned Strong Coupling Algorithms for Fluid–structure Interaction*. Comput Struct. 2003;81(8–11):805–12.

138. Dettmer W., Perić D. *A Computational Framework for Fluid–structure Interaction: Finite Element Formulation and Applications*. Comput Methods Appl Mech Eng. 2006;195:5754–79.

139. Dettmer W., Perić D. *A Fully Implicit Computational Strategy for Strongly Coupled Fluid–Solid Interaction*. Arch Comput Methods Eng. 2007;14(3):205–47.

140. Hübner B., Walhorn E., Dinkler D. *A Monolithic Approach to Fluid–structure Interaction Using Space–time Finite Elements*. Comput Methods Appl Mech Eng. 2004;193(23–26):2087–104.

141. Gee M. W., Küttler U., Wall W. A. *Truly Monolithic Algebraic Multigrid for Fluid–structure Interaction*. Int J Numer Methods Eng. 2011;85(8):987–1016.

142. Greenshields C. J., Weller H. G. *A Unified Formulation for Continuum Mechanics Applied to Fluid–structure Interaction in Flexible Tubes*. Int J Numer Methods Eng. 2005;64(12):1575–93.

143. Jameson A. *Transonic Flow Calculation: MAE Report #1651*. 2003.

144. Doetsch G. *Guide to the Applications of the Laplace and Z Transforms*. Second Edi. London, UK: Van Nostrand Reinhold Company; 1971.

145. Juang J.-N., Suzuki H. *An Eigensystem Realization Algorithm in Frequency Domain for Modal Parameter Identification*. J Vib Acoust Stress Reliab Des. 1986;110(1):24–9.

146. Silva W. A. *Identification of Nonlinear Aeroelastic Systems Based on the Volterra Theory: Progress and Opportunities*. Nonlinear Dyn. 2005;39:25–62.

147. Griffiths L. M., Gaitonde A. L., Jones D. P., Friswell M. I. *Updating of Aerodynamic Reduced Order Models Generated Using Computational Fluid Dynamics*. Proc Inst Mech Eng Part G J Aerosp Eng. 2017;232(9):1739–63.

148. Simeone S., Rendall T., Williams S. P. I., Wales C., Cooper J., Jones D., et al. *Reconstruction of Gust Velocity Profiles via Potential Flow, CFD and ROM Techniques*. In: 17th International Forum on Aeroelasticity and Structural Dynamics (IFASD). Como, Italy; 2017. p. 1–18.